

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to the provisional patent application with the following Serial Number: 60/222,827 filed on August 3, 2000.

TECHNICAL FIELD

The present claimed invention relates to the field of wireless communication.

15

BACKGROUND OF THE INVENTION

Wireless communication has extensive applications in consumer and business markets. Among the many communication applications/systems are: fixed wireless, unlicensed (FCC) wireless, local area network (LAN), cordless telephony, personal base station, telemetry, mobile wireless, and other digital data processing applications. These applications generally utilize unique and incompatible protocols for various signal processing operations, e.g., encoding, modulation, demodulation, and decoding, etc. These unique and incompatible protocols may require unique hardware, software, and methodologies for the communication protocol. This practice can be costly in terms of design, testing, manufacturing, and infrastructure resources. As a result, a need arises to overcome the limitations associated with the varied hardware, software, and methodology for processing digital signals in each of the varied wireless applications.

In contrast to the hardware and algorithmic variations in wireless applications, they all share a common demand for increased capacity to accommodate new users that continues to grow at an enormous rate. Compounding this problem is the demand for new and more data-intensive forms of wireless communication, such as data transfer with networks, e.g., Internet data transmission. In contrast, the resources available to accommodate these demands, e.g., frequency bandwidth, are substantially limited. Consequently, a need arises for an apparatus and a method to effectively accommodate the increases in the quantity of users and the increase in the quantity of data transferred while using a limited frequency bandwidth.

Besides the variation between communication applications, substantial variations

5 occur over time within a given communication application. For example, within the time
division multiple access (TDMA) cellular wireless application, there have been a
proliferation of different versions and performance levels, e.g., Telecommunication Industry
Association (TIA) Interim Standard-54B (IS-54B), IS-136, Global System for Mobile
10 Communications (GSM), GPRS, EDGE, etc. And the pace at which improvements and new
standards arise is increasing as more industry resources are focused on the needs and
opportunities in wireless communication. Unfortunately, all these factors result in minimal
uniformity around the world at any one given point in time. For example, different countries
and different service providers frequently use systems that are uniquely dedicated to their
specific version of a communication protocol. Consequently, a need arises for overcoming
15 the limitations of protocol nonuniformity and proliferation within each of the wireless
communication applications.

The proliferation of communication protocols generates yet other problems. For
example, the cost of changing communication protocols or upgrading versions or
performance levels within a communication protocol can be significant. That is, handset and
20 base station designers frequently improve the signal processing algorithms and processes to
improve service. Given the high quantity of base stations, as well as user handsets, even a
small unit cost for a change can multiply into a very large cost for the entire system. These
costs are most pronounced when a hardware change or when on-site field reprogramming is
required. Furthermore, a software or hardware change for a new version or performance
25 level may hinder the efficiency of the existing device configuration due to incompatibility,
etc. Consequently, a need arises to overcome the limitations of cost and resource intensive
changes in versions or performance levels of a communication protocol.

Changes in performance level or versions of a communication protocol can also affect
the network services and coverage, and hence the survival of a wireless service provider or a
30 hardware manufacturer. For example, given the long lead time and the investment required
for designing, manufacturing, and installing an infrastructure for a given communication
protocol, a future but uncertain specification can be a tremendous risk. This is especially so
with an application specific integrated circuit (ASIC) device whose configuration is defined
primarily by fixed hardware. However, market rewards may be significant for the service
35 provider or manufacture that is able to realize the new protocol in the shortest possible time.

5 Thus, a risk versus reward tradeoff exists with implementing new communication protocols. Given the degree of the risks and promise of rewards, a need arises to overcome the limitations of the long lead-time and the investment required for implementing a new specification.

With the increased sophistication of each new generation of communication device, power consumption remains a significant issue. Among other things, power consumption affects: battery life for handheld devices; cooling systems required for base stations; durability and reliability of semiconductor devices and integrated circuits; and other performance criteria. Conventional alternatives to hardwired ASICs have significant power consumption issues that offset their benefits. Consequently, a need arises to overcome the limitations in power consumption for a communication device.

SUMMARY OF THE INVENTION

The present invention provides a method and apparatus that can effectively accommodate the increases in the quantity of users and the quantity of data transferred using the limited frequency bandwidth. Furthermore the present invention provides a solution that overcomes the limitations of protocol nonuniformity and proliferation in the wireless communications. The limitations of cost and burden associated with changes in versions or performance levels of a communication protocol are also resolved by the present invention. In an effort to minimize the risks and maximize the rewards, the present invention substantially shortens the lead-time and the investment required for implementing a new specification. Finally, the present invention provides reasonable power consumption for the communication device.

In particular, the present invention provides an apparatus and method that can flexibly and efficiently process data. One embodiment of the present invention is a processor with an architecture that includes a first computing element, a second computing element, and a reconfigurable interconnect. The first computing element is coupled to the second computing element via the reconfigurable interconnect. The first computing element performs a discrete operation, or portion thereof in an application, while the second computing element performs another discrete operation, or portion thereof for the application. The reconfigurable interconnect has an uncommitted architecture, thereby allowing it to be configured by an

5 outside source to couple portions of the first reconfigurable interconnect with portions of the second reconfigurable interconnect in any combination. The first computing element, the second computing element, and the reconfigurable interconnect operable to perform a class of functions suitable for processing the communication signal.

10 A second embodiment of the present invention provides a convenient method for operating the functional kernels having reconfigurable hardware to a configuration for implementing wireless communication. In a first step of the process, a signal to be processed is received at a reconfigurable modem platform having a heterogeneous multiprocessor architecture. Next, the signal is assigned to a data pump path in the reconfigurable modem platform. Then, design configuration information for the reconfigurable modem platform is
15 received. Finally, the data portion of the signal undergoes digital signal processing using the reconfigurable modem platform.

These and other objects and advantages of the present invention will become apparent to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments, which are also illustrated in the various drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

20 The drawings included herewith are incorporated in and form a part of this specification. The drawings illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention. It should be understood that the drawings referred to in this description are not drawn to scale unless specifically noted as
25 such.

Figure 1A is a block diagram of the interface functions accommodated by the electronic communication device, in accordance with one embodiment of the present invention.

30 Figure 1B is a block diagram of an electronic communication device having heterogeneous multiprocessor components, in accordance with one embodiment of the present invention.

Figure 1C is a block diagram of another configuration of an electronic communication device having heterogeneous multiprocessor components, in accordance with one
35 embodiment of the present invention.

5 Figure 1 D is a block diagram of an electronic system used to interface a user with a configurable multiprocessor device, in accordance with one embodiment of the present invention.

Figure 2A is a block diagram of a processor having multiple configurable hardware kernel planes used in the electronic communication device, in accordance with one
10 embodiment of the present invention.

Figure 2B is a block diagram of multiple possible architecture techniques used in the algorithmic satellite kernel portion of the hardware kernel, in accordance with one embodiment of the present invention.

Figure 2C is a block diagram of a configurable hardware kernel plane, in accordance
15 with one embodiment of the present invention.

Figure 2D is a block diagram of a kernel portion of a hardware kernel plane, in accordance with one embodiment of the present invention.

Figure 2E is a block diagram of a hardware kernel containing a subcomponent hardware kernel, in accordance with one embodiment of the present invention.

Figure 2F is a block diagram of the inputs, outputs, and functions accommodated by the algorithmic satellite kernel in the electronic communication device, in accordance with
20 one embodiment of the present invention.

Figure 3 is a block diagram of the modem functions accommodated by the electronic communication device, in accordance with one embodiment of the present invention.

Figure 4 is a block diagram of the codec functions accommodated by the electronic communication device, in accordance with one embodiment of the present invention.
25

Figure 5 is a diagram of the separation and combining process of a single thread into multiple concurrent threads to be accommodated on the communication device, in accordance with one embodiment of the present invention.

Figure 6A is a flowchart of the process used to implement a design configuration onto a configurable electronic communication device, in accordance with one embodiment of the present invention.
30

Figure 6B is a flowchart of the process used to operate a configurable electronic communication device, in accordance with one embodiment of the present invention.
35

DETAILED DESCRIPTION OF THE INVENTION

Reference will now be made in detail to the preferred embodiments of the invention.

Examples of the preferred embodiment are illustrated in the accompanying drawings. While the invention will be described in conjunction with the preferred embodiments, it is understood that they are not intended to limit the invention to these embodiments. Rather, the invention is intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope of the invention, as defined by the appended claims. Additionally, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one of ordinary skill in the art that the present invention may be practiced without these specific details. In other instances, well-known methods, procedures, components, and circuits have not been described in detail so as not to unnecessarily obscure aspects of the present invention.

The present invention can be implemented in a wide variety of digital wireless communication systems or techniques that utilize code sequences. Code sequences are utilized in wireless communications for many functions including, but not limited to: filtering, searching, modulation, and demodulation. The systems or techniques which utilize code sequences include, but are not limited to, fixed wireless, unlicensed Federal Communications Commission (FCC) wireless systems, wireless local area network (W-LAN), cordless telephony, cellular telephony, personal base station, telemetry, and other digital data processing applications. The present invention can be applied to both transmitters, e.g., a base station, and to receivers, e.g., a terminal, for fixed wireless, W-LAN, cellular telephony, and personal base station applications.

In particular, one fixed wireless application to which the present invention may be applied is a metropolitan multipoint distribution system (MMDS). Examples include wireless cable broadcast, or two-way wireless local loop (WLL) systems. Some examples of a W-LAN, that can communicate digitized audio and data packets, for which the present invention can be applied include Open Air, and the Institute of Electrical and Electronics Engineers (IEEE) specification 802.11b. In yet another application, a specific example of unlicensed FCC applications to which the present invention may be applied includes the Industrial, Scientific, and Medical band (ISM) devices, which can include cordless telephony

5 products. Personal base stations can utilize either cordless or cellular telephony wireless
communication standards. Lastly, the cellular telephony systems in which the present
invention can be applied are time division multiple access (TDMA) systems including, but
not limited to Global System for Mobile Communications (GSM), GPRS, EDGE, IS-54B, IS-
136. The range of code sequences utilized in the exemplary applications disclosed herein is
10 useful to define the class of functions for which the present configurable code generator unit
is applicable.

The detailed description of the present invention begins with a description of an
exemplary application, a spread-spectrum communication device, in Figures 1A through 1D.
In particular, communication device is described in Figure 1A in terms of a functional block
15 diagram, in Figures 1B and 1C in terms of hardware block diagrams, and in Figure 1D as a
block diagram of a virtual machine interface (VMI) that translates user-desired functions to
device-level instructions using resident hardware and software. Next, Figures 2A through 2F
describe a configurable processor, and its configurable hardware kernel components, in an
increasingly detailed manner. A functional block diagram of a configurable hardware kernel
20 is presented in Figure 2F. Thereafter, the functional layout of hardware kernels in a
configurable modem processor for the exemplary communication device is provided in Figure
3 for modem functions and in Figure 4 for codec functions. Figure 5 provides a diagram of
function management technique utilized by processors with configurable hardware kernels.
Lastly, various processes associated with the communication device and the hardware kernels
25 are described in Figures 6A and 6B.

COMMUNICATION DEVICE

Referring now to Figure 1A, a block diagram 10 of the interface functions
accommodated by the electronic communication device are shown, in accordance with one
30 embodiment of the present invention. The signal processing functions 16 shown have a
configurable architecture in one embodiment that enables an electronic communication
device to operate using a wide variety of communication protocols, including TDMA, as well
as anticipated future protocols. Block diagram 10 provides a macro level description of the
interface functions. A more detailed description of the specific sub-functions and operations,
35 and the hardware that implements them are provided in subsequent figures herein.

5 The signal processing function block 16 of a communication device is responsible for performing the data processing steps needed to convert a received signal into meaningful data for an end user and to convert an input from a user and convert it to a transmittable signal. Within signal processing function block 16, a wide variety of functions, sub functions, and operations are performed to satiate the requirements of the specific communication protocol
10 chosen for a communication device. For example signal-processing functions can include demodulation and/or decoding for received in-phase and quadrature-phase (I/Q) samples 12, and include encoding and/or modulation for transmitted I/Q samples 14. Control data 20 provides the necessary control information, e.g., flags, handshakes, addressing, status, etc., to components such that signal processing functions 16 can be accomplished. Signal processing
15 function block 16 also communicates with a traffic interface 18, to accommodate the allocation of respective signals to each of the many different wireless users operating within a given communication system. In the present embodiment, the traffic interface is a mobile telephone switching office (MTSO) for a cellular TDMA communication application. However, other TDMA communication application may not require a traffic interface, e.g., a cordless telephony application. Processing function block 16 receives I/Q samples 12 upon which known data processing protocols are implemented. Similarly, processing function block 16 transmits desired data as I/Q samples 14 to another communication device (not shown). Configuration data 11 inputs provides the necessary information to configure the hardware and software components used to implement signal-processing function block 16.

20 Referring now to Figure 1B, a block diagram of an electronic communication device having heterogeneous multiprocessor components for processing data is shown, in accordance with one embodiment of the present invention. Electronic communication device 100 includes subcomponents and configurations that are illustrated and described in more detail in subsequent figures. Additionally, electronic communication device 100 also
25 employs methodology for design, configuration, and operation that will be described in subsequent flowchart figures. While electronic communication device 100 is a base transceiver station (BTS) in the present embodiment, the present invention is well suited to electronic communication device 100 being a mobile handset or a test equipment platform.

30 Electronic communication device 100 includes an interface conversion block 116, a modulator/demodulator (modem) processor I 02a, an optional configurable modem processor

102b, memory 106 and 118, a processor 112, a channel coder/decoder (codec) processor 104, a base transceiver station (BTS) card controller 110a, and an ATM Utopia/HDLC 108.

Processor 112 can either be a digital signal processor (DSP) or general-purpose microprocessor (GP uP). External memory 106 used for interleaving meets the following requirements for the present embodiment: 1) 8 Mb SRAM; 2) 18 MHz Performance; 3) Minimum 512K x 16 configuration; 4) Byte write-enables. However the present invention is well suited to alternative memory configurations, tailored for a given application.

In the present embodiment, configurable modem processor 102a, optional configurable modem processor 102b, and configurable channel codec processor 104 have a configurable heterogeneous multiprocessor architecture. Thus, configurable modem processor block 102a and configurable codec processor 104 can be configured to operate a wide range of communication protocols, e.g., the exemplary protocols described hereinabove. To do so, configurable modem processor 102a and configurable codec processor 104 are designed with a sufficiently wide scope to accommodate the common and unique requirements of these varied communication protocols. Subsequently, the configurable modem processor 102a and the configurable codec processor 104 are provided with the instructions and data necessary to configure them, e.g., configuration input 121, to a single desired TDMA protocol. Notably, the configurable modem processor 102a and the configurable codec processor 104 can subsequently be reconfigured to another TDMA protocol within the design scope. The hardware, software, and processes used to implement this configuration are described in subsequent figures. The specific structure, components, functions, and processes utilized for the present invention will be described in more detail in subsequent hardware, function, and flowchart diagrams. This architecture provides electronic communication device 100 with a flexible configuration that can accommodate a wide range of communication applications while simultaneously providing energy efficient operation. Thus the present invention overcomes the trade-offs associated with prior art configurations, wherein a device was flexible but energy inefficient, or the device was energy inefficient but inflexible.

Configurable modem processor block 102a includes at least one hardware kernel plane in the present embodiment. The hardware kernel plane, described in subsequent figures, is a reconfigurable collection of multiple algorithmic-specific processors. The

5 hardware kernel plane can implement the sub functions of a functional kernel plane,
described in a subsequent figure. In the present embodiment, configurable modem processor
block 102a can be designed to accommodate multiple channels of data, the specific number
depending upon the needs of a specific application.

Channel codec functions are implemented by exemplary CODEC signal processor 104
10 of Figure 1B, which is a reconfigurable multi-channel digital channel encoder-decoder chip
that performs convolution, iterative turbo decoding, rate-detection, and rate matching
functions for a wide range of quantities of voice channels. Channel codec functions also
implemented by channel codec signal processor 104 for transmission encoding include
convolution and turbo coder functions, puncturing, rate-matching, and interleaving on the
15 transmit path, in the present embodiment.

By using a heterogeneous reconfigurable architecture together with application-
specific kernels, as described hereinafter, a programmable and configurable signal-processing
platform is realized. These configurations are completely under the control of the
communication systems designer, and are described in detail in a subsequent flowchart. The
20 architecture of the channel codec signal processor 104 of Figure 1B provides a very high level
of integration, while allowing the communication systems designer to employ proprietary
techniques to improve decoder performance in each of the stages described above. The
architecture of the channel codec signal processor 104 enables the system designer to
program the chip at many levels, including control of specific datapaths to realize different
25 algorithms. Details on the architecture and operation of codec processor are given in
subsequent block diagram figures and in flowchart figures.

Interface conversion block 116 is coupled to the antenna bus 129, which is in turn
coupled to an antenna, e.g., a BTS antenna 120, which can include multiple antennae in one
embodiment. Interface conversion block 116 is also coupled to configurable modem
30 processor 102a and to optional configurable modem processor 102b via bus 122, respectively.
Bus 122 can be separate independent non-shared buses, or a single shared bus. Interface
conversion block 116 includes components (not shown) such as a radio frequency (RF)
transceiver and an analog to digital (A/D) converter coupled to each other in series, whose
subcomponents and functions are known to those skilled in the art. Configurable modem
35 processor 102a and optional configurable modem processor 102b are coupled to a processor

112 and to memory 106, which is random access memory (RAM) in the present embodiment. Memory 106 is coupled to channel codec processor 104, which in turn is coupled to ATM Utopia/HDLC 108 via bus 124. An internal data/control bus 126 is coupled to interface conversion block 116, memory 118, processor 112, BTS card controller 110a, and to ATM Utopia/HDLC 108 in order to pass data and control instructions between these components. ATM Utopia/HDLC 108 is a conventional asynchronous transfer mode (ATM) networking interface in the present embodiment.

In one embodiment, uP 112 is a general-purpose microprocessor capable of performing the functions of BTS card controller 110a. An external BTS cell controller 114 is coupled to internal bus 126 and to a mobile telephone switching office bus 128. ATM Utopia/HDLC 108 provides signal formatting, memory, and interface circuits for Operations and Maintenance (OAM) control. BTS Cell Controller 114 determines the personality (or radio configuration) of each channel element by loading specific configuration software into specific control registers in BTS Channel Card Controller 110a, processor 112, configurable modem processor 102a, and codec processor 104. Interface configurations and protocols are described in Figure 1C.

By using processor 112 in communication device 100, the present invention provides the ability to use existing digital signal processing resources, while upgrading other processing resources to more flexible and efficient hardware, e.g., configurable modem processor 102a. In particular, functions that were performed on general-purpose processors are performed by the present invention on operation-specific, or algorithmic-specific, processors that are interconnected to realize a modem or codec function. Furthermore, the operation-specific processors possess the appropriate amount of configurability for protocol variations, thereby providing robustness for future developments. By providing configurability, the present invention provides a communication device with the flexibility to operate under a wide range of TDMA communication protocols.

Electronic communication device 100 is responsible for processing voice, data, and control signals in a transmission and reception mode. In pursuit of these functions, BTS card controller 110a provides a number of interfaces. In the present embodiment, BTS card controller 110a provides: 1) antenna receive (RX) Bus 129 to receive the uplink digitized signal samples; 2) BTS Cell Controller for sending and receiving control information

5 associated with call setup, teardown, and handoff; and 3) Operations and Maintenance (OAM) Monitor for performance analysis, reconfiguration over the network based on system planning, and troubleshooting.

Alternative components and configurations to communication device 100 are compatible with the present invention. For example, bus 126 provides an exemplary coupling
10 configuration of components in electronic communication device 100. It is appreciated by those skilled in the art that bus 126 can include subcomponents of specific control/status/data lines for communication between appropriate devices. It is further appreciated by those skilled in the art that bus 126 can be a parallel configuration or serial configuration with multiplexing. Furthermore, bus 126 can have interconnects and translators, as appropriate for
15 a given application. These alternatives are also suitable for buses 129, 128, 127, 130, 122, and 122a, and other buses that can be used to couple components in communication devices 100 and 101 of Figures 1B and 1C, respectively.

Additionally, communication device 100 is well suited to alternative components and coupling configurations of memory 106. For example, while memory 106 is located between
20 configurable modem processor 102a, and channel codec processor 104 in the present embodiment, the present invention is well suited to coupling configurable modem processor 102a directly to channel codec processor 104 and to locating memory 106 directly adjacent to channel codec processor 104. While memory 106 and 118 are specified as RAM in the present embodiment, the present invention is well suited to a wide range of memory
25 configurations, such as ROM, registers, flash memory, etc. While antenna 120 is shown as a BTS antenna having multiple individual antennae arranged in sectors, the present invention is well suited to antenna 120 be a single or dual antennae for a mobile handset or a test platform application.

While the present embodiment provides both a configurable modem processor 102a
30 and a configurable codec processor 104 in communication device 100, the present invention is well suited to an alternative configuration that uses a configurable modem processor 102a with a conventional codec processor (e.g., a digital signal processor), and to using a configurable codec processor 104 with a conventional configurable modem processor. Lastly, while the present invention provides a single modem processor, with a single optional
35 configurable modem processor 102b, the present invention is modular, and thus is well suited

5 to using a wide range of processor types and quantities, as appropriate for a given application. Communication device can also include a configuration that accommodates multiple communication standards. For example, the present invention can be applied to communication protocols such as orthogonal frequency division multiplexing (OFDM). More detail is provided in a commonly assigned and related application, which is incorporated
10 herein by reference and entitled "METHOD AND APPARATUS TO SUPPORT MULTI STANDARD, MULTI SERVICE BASE-STATIONS FOR WIRELESS VOICE AND DATA NETWORKS," Attorney Docket No. 9824-0035-999, US patent application serial number 09/752,050, filed on December 29, 2000.

Referring now to Figure 1C, a block diagram of another configuration of an electronic
15 communication device having heterogeneous multiprocessor components is shown, in accordance with another embodiment of the present invention. Second configuration 101 of an electronic communication device has many components and configurations that are similar to those shown in Figure 1B. Thus, components and coupling configurations different from Figure 1C are primarily discussed hereinafter.

20 Communication device 101 includes a BTS card controller 110b that can be a state machine or an optional microprocessor. Bus 127 couples memory 118 and BTS card controller 110b to channel codec processor 104 and configurable modem processor 102a. This provides a more direct data route between memory 118 and configurable modem processor 102a and channel codec processor 104. Additionally, memory 106 is located
25 adjacent to channel codec processor 104, and coupled to BTS card controller 110b, and provides improved communication and processing to channel codec processor 104. It also provides improved communication and processing between channel codec processor 104 and modem processor 102a. While only one configurable modem processor 102a is shown in Figure 1C, communication device 101 is well suited to using more than one configurable
30 modem processor, as appropriate in a given application. Communication device 101 does not have a separate conventional DSP chip like communication device 100. Rather, communication device 101 utilizes either an external general purpose microprocessor 103 or utilizes computing elements within configurable modem processor 102a and channel codec processor 104 to perform functions traditionally provided by a conventional DSP chip. The
35 alternative configurations and components discussed for communication device 100 are

5 likewise applicable to communication device 101.

Interface configurations and protocol are utilized between components within communication device 100 and 101 as well as between components within and outside of communication device 100 and 101. For example, a bus interface 130a and 131 of communication device 100 and 101, respectively, is provided for streaming the received, 10 coded symbols from the modem signal processor 102a and/or 102b to the channel codec signal processor 104. Similarly, bus interface 131 is provided for streaming the encoded transmit data from channel codec signal processor 104 into the modem signal processor 102a-102b (assumed to be interleaved on channel codec signal processor). Busses 130a and 130b of Figure 1B and bus 131 of Figure 1C combine a hand-shaking mechanism with a well- 15 defined data stream in order to provide necessary flexibility in programming. Interface conversion/sector combining block 116 provides a high-speed parallel interface for communicating digitized I/Q samples between the modem signal processors 102a and 102b, and antenna 120, for an uplink or downlink embodiment. Two parallel ports 122 (one for I/Q uplink, one for I/Q downlink) are provided for multiplexed interface to multiple antennae, 20 e.g. in antenna array 120 for a base station in the present embodiment. Memory interface 132 is provided for interfacing to an external SRAM 106 for support of necessary deinterleaving required for high data rate support. The interlace assumes that a single memory resource may be shared across multiple modem signal processor implementations 102a (up to three modem signal processor's per memory), so that semaphores are required for coordinating, via input 25 134 from BTS card controller 110b, the necessary memory writes and establishing a single modem signal processor as the burst controller for frame bursts out of the deinterleaver memory. A well-defined memory map within memory 106 is used to avoid fragmentation across external memory 106.

The functions and interlaces shown in Figure 1A are implemented, in one 30 embodiment, using hardware shown in Figures 1B-1C, and using subcomponents described in subsequent figures. For example, received I/Q samples 12 are provided to signal processing function block 16, via bus 129 of Figure 1B in the present embodiment. Similarly, the signal processing functions required for an application are performed by communication device 100 of Figure 1B or device 101 of Figure 1C. Thus, for example, transmitted I/Q samples 14 and 35 received I/Q samples can be communicated via bus 129 to/from antenna 120. Traffic

5 interface 18 is accommodated via bus 128, and control data 20 is accommodated via bus
126, in one embodiment. While the functions described are attributed to a base transceiver
station (BTS), the present invention is well suited to implementing the functions and
appropriate interfaces on a mobile handset. For example, a mobile handset would not have an
MTSO traffic interface, but would include the balance of the interfaces and functionality of
10 Figure 1A. While the present embodiment only describes configurable processors for two
function types, e.g., modem and codec functions, and applies them to a wireless
communication application, the present invention is well suited to accommodating other
functions for other applications in a similar manner.

Referring now to Figure 1D, a functional block diagram 150 of a system used to
15 interface a host-processor containing a user's program of configurations with a configurable
multiprocessor device is shown, in accordance with one embodiment of the present invention.
The configurable components, e.g., hardware kernels and interconnect of Figure 2C, for
communication device 100 can be programmed by user-defined configurations. The user-
defined configurations can be generated by a user from a host microprocessor, e.g., a
20 workstation, which implements a programming interface (API).

Processor hardware (HW) model 160 is a block diagram representing the interaction
of hardware, e.g., a processor 112, with software and data such as library functions,
instructions, and configuration data, that are stored in memory. Processor software model can
be programmed with a virtual machine interface (VMI) in one embodiment. Specifically,
25 processor 112 can be modeled as a software model consisting of user's C-language code (or
software) 164 and I/O device drivers 166. The user provided C-language software 164 in
order to configure configurable processes 170 of the configurable modem 102a and 102b, and
of the configurable coded 104. C-language software 164 includes resident user-implemented
C language block of instructions and functions that interface with input/output (I/O) driver
30 block 166. C language block 164 is coupled to receive C-based application programming
interface (API) programming guide data 171, which includes a library of functions for
programming configuration inputs 121a through 121e to appropriate configuration mappings.
Similarly, processor software model 162 is coupled to receive input 172 of API functions
mapped to instruction set 171 and 172. Each API function of input 172 includes a set of
35 instructions which will: 1) map the API function to a specific hardware kernel and

reconfigurable interconnect (e.g. DRL) process; 2) describe the DRL process as a set of data structures/configuration signals that are passed via hardware interfaces 168 by host processor 112. API Programming Guide 171 serves as a reference to access the modes of flexibility in the modem signal processor.

Hardware interfaces block 168 is coupled to processor 112 to receive device driver information appropriate for the configuration input 121a through 121e. In turn, hardware interfaces block 168 is coupled to provide configuration instructions and data, e.g., driver information, to dynamically reconfigurable logic (DRL) block 170. DRL processes block 170 represents the ultimate desired processes (or functions) to be implemented by the configurable communication device, e.g., device 100 of Figure 1B. DRL processes block can communicate configuration mapping data 174 to appropriate configurable components, described in subsequent Figures 2A through 2F, of configurable communication device 100, which can store and transmit instruction sets of mapped API functions. User-implemented C-language block 164 provides a user with the ability to control the configuration of a hardware kernel, described in subsequent Figures 2C through 2E. In particular, the programmer/user can provide the following inputs to define the configuration of a configurable hardware kernel: 1) inputs, as shown by input 121a; 2) outputs, as shown by input 121b; 3) operation(s) to be performed, as shown by input 121c; 4) parameters, as shown by input 121d; 5) a type of arithmetic in a processing unit (e.g., dataflow process); and 6) type of state machines controlling the dataflow process (e.g., controlflow process) as shown by input 121e.

In addition, user-implemented C-language block 164 provides a user with the ability to control the reconfigurable interconnect that links multiple hardware kernels within a kernel plane. Thus, input 121e includes user-specified interconnect configurations (i.e. interconnect reconfigurability). The present invention is well suited to using less inputs and control or providing additional inputs or control for a user to configure a configurable device. Interconnect configuration input 121e specifies the configuration of reconfigurable interconnect such that a cluster(s) of kernels may be built. This level of control with individual kernels and the interconnect offers substantially greater flexibility than that of hardwired or parameterized application specific integrated circuit (ASIC) solutions. By using kernels that are focused to the specific types of data processing found in terrestrial and

5 wireless communication applications, the computational efficiency, in millions of operations per milliwatt (MOPS/mW) of this architecture approaches that of an ASIC.

Functional block diagram 150 shows the hierarchy with which a user may interface DRL processes block 170. That is, the hierarchy includes: a) a first level with a C-based interface 164 with a user, b) a second level with an I/O device driver 166 interface between
10 the C-language and the hardware interfaces; c) a third level with hardware interface block 168 that interfaces between the DRL process 170 and the processor HW model 160. Thus, the present embodiment utilizes a hierarchical Application Programming Interface (API) that supports several levels of user control for programming a Channel Codec Signal Processor, e.g., codec processor 104, or a configurable modem processor 102a, as shown in Figure 1B.

15 Dashed box 176 Includes the API functions mapped to instruction set data 172, and the configuration mappings data 174. Dashed block 176 indicates the control software, which includes a subset of functions defined by the API. The control software block 176 enables the function of the configurable communication device, e.g., device 100 of Figure 1B.

C-based API programming guide 171 utilizes a mechanism referred to as extensible data types (EDT), described in more detail in a subsequent flowchart. The use of EDT effectively
20 embeds a mechanism in the API that allows additional functionality to be transparently added, the API is able to add new hardware services without the need to modify existing software. This mechanism must also provide this abstraction without significant performance overhead. By the use of pointer access to data types and inline functions, this effect is
25 minimized. By abstracting the hardware from the software, the API allows the hardware and software development processes to be decoupled from one another. This significantly reduces development time, as each hardware upgrade or modification does not require changes to the software. Modes of configurations are specified in the API programmer's guide 171 for hardware kernel blocks. API programmer's guide 171 describes the allowed dataflow
30 configurations (i.e. hierarchical interconnect configurations) of the machine.

Through the use of extensible data types, a hardware configuration can be constructed to support various cellular telephony standards. Furthermore, the types can be configured to support various performance requirements for the system as a whole, for a set of mobiles, or for a particular mobile. Hardware configurations are realized through an API that allows a
35 programmer to manipulate the relationships between various extensible data types. The

resources available in an extensible data type directly map to available hardware resources. A programmer using the API can configure the hardware resources from a conceptual viewpoint.

The programmer's model and API (or VMI) of Figure 1D defines and provides a mechanism for a user to systematically and efficiently develop software that controls the function and operation of the configurable communication device, e.g., 100 of Figure 1B. The API allows the programmer to manage the hardware resources without the need to write complex low-level hardware-target-dependent code. This provides many advantages including ease of adoption and integration of the configurable communication device. Subsequent flowcharts provide more detailed description of the process for developing the software to control the configurable communication device.

By providing a high level API, a user can design his software in a top-down fashion. This enables top-level system problems to be rapidly identified and corrected before the low-level code is written. Additionally, this approach saves a significant amount of development time as it removes the need to rework low-level software to match high level changes. The programmer's model and API of Figure 1D also provides efficient use of hardware parallelism. Thus, the present invention provides a method and architecture that overcomes the challenging task of scheduling the many hardware resources in the complete system. This requires an efficient mechanism for communication between the hardware resources, both within a configurable processor, e.g., within configurable modem processor 102a of Figure 1B, and between the configurable processors (e.g., between configurable modem processor 102a/codec processor 104 and the controlling processors, e.g., processor 112, BTS card controller 110a or 110b, and BTS cell controller 114 as shown in Figures 1B and 1C). The hardware utilization, scheduling, and maintenance are under the control of the API. By embedding these mechanisms in the API, a process can be designed in isolation, with the synchronization issues handled at only one level within the software hierarchy. This produces a system that is considerably quicker to build and more efficient in the use of hardware than one that uses many synchronization techniques within the design. Additional description on the process for providing C-based API programming guide 171 and API functions mapped to instruction set 172 is provided in co-pending patent application entitled "A METHOD FOR DESIGNING A CONFIGURATION FOR A CONFIGURABLE SPREAD SPECTRUM

5 COMMUNICATION DEVICE,” Attorney Docket No. 9824-083-999, US patent application serial number 09/772,582, filed on January 29, 2001. This related application is commonly assigned, and is hereby incorporated by reference.

10 The present embodiment of Figure 1D can include additional features for individual versions of the software. Examples of this would be an input data formatter to deal with the incoming data format on a particular test rig, or the creation of a software version that performs function profiling or debugging on a particular section of the software, without having the attendant performance degradation on other blocks. Both of these alternatives are dealt with efficiently by the use of extensible data types, which allow a user to rapidly modify the functionality of the software by modifying the data types and recompiling.

15 HARDWARE KERNELS

Referring now to Figure 2A, a block diagram of a configurable modem processor 102a having multiple configurable hardware kernel planes used in an electronic communication device is shown, in accordance with one embodiment of the present invention. Figure 2A illustrates how the multiple hardware kernel planes interconnect and shows what device controls and coordinates them. Processor 102a can be configured to operate as a configurable modem processor 102a and 102b or as a configurable codec processor 104, as shown in Figures 1B and 1C, in the present embodiment.

20 In the present embodiment, two hardware kernel planes, e.g., plane [1]201a and plane [i] 201i, are coupled to an allocator 219 via a reconfiguration bus, e.g., 206a and 206i, respectively. Each hardware kernel plane is assigned to a given channel in a communication system in the present embodiment. In turn, allocator 219 is coupled to a general-purpose (GP) microprocessor or signal processor 112. To reduce overhead in terms of instruction fetch and global control, the architecture utilizes distributed control and configuration. The communication mechanism between each kernel is dataflow driven. Allocator 219 performs controller operations for each of the kernel planes 201 a and 201i, such that they can operate independent of each other, e.g., in parallel.

25 To perform these functions, allocator 219 includes memory and state machine components. In one embodiment, allocator 219 is configurable such that it can manage the desired type of functions implemented in the hardware kernel planes 201a through 201i. For

5 example, allocator 219 can be configured such that hardware kernel planes 201 a through
201i perform modem functions, or alternatively codec functions. While the present
embodiment shows only two hardware kernel planes 201a and 201i, the present invention is
well suited to using any quantity of hardware kernel planes in processor 102a. Hardware
kernel planes 201a and 201i include a configurable heterogeneous multiprocessor architecture
10 that will be described in more detail hereinafter. Data bus 122 provides data to and from
configurable modem processor 102a, as does data line 130a, as shown in Figure 1B.

Processor 112 performs higher-level management operations for the allocator. In this
manner, multiple instances of processor 102a can be accommodated in a communication
device, and can operate in varying degrees of parallel processing. In the present embodiment,
15 processor 112 is a system processor of the parent communication device 100 of Figure 1B.
However, configurable processor 102a can have a dedicated local microprocessor in lieu of
system processor 112.

Referring now to Figure 2B, a block diagram 200b of multiple possible architecture
formats used in a hardware kernel portion of a communication device is shown, in accordance
with one embodiment of the present invention. By using multiple levels of granularity in its
components, the communication device possesses a wide breadth of efficient
programmability. And efficient programmability translates into accommodating of multiple
non-uniform specifications by a communication device with each level of granularity having
its own preferred target for a given application. A systematic and hierarchical method to
20 exploit the flexibility of incorporating these different architectures into hardware is described
in a subsequent flowchart figure.

Figure 2B shows the four main levels of programmable or reconfigurable granularity
used in a hardware kernel in the present embodiment. The difference in the various
computational models shown in Figure 2B lies in the granularity of the composing modules,
the distribution of the program storage, and the interconnect structure. In one embodiment,
30 the computing elements in a hardware kernel can exploit any combination of the four types of
reconfigurability, in an architecture referred to as Dynamically Reconfigurable Logic (DRL),
described in more detail hereinafter. However, the present invention is well suited to
incorporating other types of computational models that have different levels of granularity or
35 different applications of granularity.

5 A first architecture format is referred to as reconfigurable logic 211. Reconfigurable logic 211 uses multiple processing islands, also referred to as configurable logic blocks (CLB), e.g., 210a coupled by an interconnect 214 with reconfigurability via bus lines, e.g., 212a, 212b, 213a, and 213b. The reconfigurable logic type of engine relies almost exclusively on bit-level mesh networks in the present embodiment. In the present embodiment, 10 interconnect 214 provides all possible coupling arrangements between the bus of data lines 212a and 212b. In this manner, independent blocks 210a -210d can communicate with one another in any desired manner. That is, they are not restricted to communicating with less than all existing kernels due to limited hardware wiring. In another embodiment, interconnect 214 can provide only a limited amount of interconnectability, based upon 15 perceived needs and capabilities of each kernel for a given application. Reconfigurable logic 211 uses bit-level operations such as encoding. By itself, reconfigurable logic provides significant benefits of flexibility. However, the flexibility comes at a trade-off of inefficiency in chip area and in power consumption. In one embodiment, processing islands have unrestricted reconfigurability of their component logic devices.

20 A second architecture format is referred to as reconfigurable datapath 221. The interconnect network of the reconfigurable datapath exploits the bit-sliced structure and predominantly one-dimensional flow of data by using asymmetric network-reconfigurable buses in one direction and bit-level mesh in the other direction. That is, reconfigurable datapath 221 uses dedicated datapaths to transmit data between electronic components, such as mux 220 and adder 226. For example, multiplex (Mux) block 220 can multiplex data from 25 multiple data lines onto a single data line, thus changing the data path. Additionally, data may be directed along one of multiple paths to an appropriate storage register, e.g., register 0 (Reg0) or register 1 (Reg1). From an appropriate storage register, data may be directed along a path to a given function block, e.g., adder 226 or buffer 228. Reconfigurable datapath 221 30 can efficiently move data, but it lacks flexibility that is not built into the original architecture. Thus, for example, the data path is limited to the data lines built between components, e.g., 220 through 228.

35 A third architecture format is referred to as reconfigurable dataflow 231. With reconfigurable dataflow, control exists over the type of arithmetic used in a processing unit (i.e. dataflow process). The reconfigurable dataflow architecture uses a program and data bus

that feeds data and control instructions to a computation unit. In particular, block 232a and 232b generate addresses to get data from memory, e.g. 234a and 234b, to be sent to a multiply--accumulate (MAC) block 236 for processing.

A fourth architecture format is referred to as reconfigurable logic 241. Reconfigurable logic 241 refers to a real-time operating system (RTOS) where the outside source controls the type of state machines that control the dataflow process (i.e. controlflow process). With reconfigurable logic 241, the stored-instruction engines rely on shared buses for the transfer of data and instructions. Block 240a is the data memory storage of data to be processed, while block 242 is the program memory for storing program instructions used to run on instruction decoder and controller 246. Block 244 is the datapath block, which contains the arithmetic operations for processing the data. Memory block 240b is a second bank of data memory for interfacing data with data path block 244.

By combining these four types of architecture, as described hereinafter, in a manner that matches the programming, function, or temporal granularity needed for a given algorithm, function, application, and/or classes thereof, the present invention provides a hybrid architecture and system. This hybrid architecture and system provides substantial improvements in performance over previously irreconcilable tradeoffs of speed, flexibility, and efficiency.

Referring now to Figure 2C, a block diagram of a hardware kernel plane used in the electronic communication device is shown, in accordance with one embodiment of the present invention. Hardware kernel plane 201a provides the capability of reconfigurability for a range of protocols in an application, or within a range of applications, with an efficiency that challenges conventional circuits. Additionally, hardware kernel plane 201a is modular, and thus may be designed to operate in groups.

Kernel plane 201a includes multiple hardware kernels K1 261a through K6 266a that are coupled to a reconfigurable interconnect 204a. Data is passed between kernels K1 261 a through K6 266a via reconfigurable interconnect 204a. Control information, such as handshake protocol signals, can also be routed through reconfigurable interconnect 204a. Hardware kernel, e.g., K1 261a, is described in detail in a following figure. Interconnect architecture supports sufficient concurrently within each of the hardware kernels K1 261a through K6 266a. In the present embodiment, reconfigurable interconnect 204a utilizes a

5 hierarchical structure that can support the required interconnect patterns (as described by the dataflow in following flowchart figures), as well as provide good performance and energy efficiency for every configuration. While the present embodiment uses six hardware kernels, the present invention is well suited to using any quantity of kernels in kernel plane 201a.

10 In the present embodiment, hardware kernels K1 261a through K6 266a kernels are specific to the types of data processing found in wireless communication applications. However, at the same time, hardware kernels K1 261a through K6 266a are heterogeneous with respect to one or more of each other, in terms of programmability, algorithmic-capability, performance-level, and/or math-logic. However, two or more kernels within
15 kernel plane 201a can be homogeneous with respect to each in another embodiment. The specific composition and relationship between hardware kernels depends upon the specific application. Examples of these levels of programmability are provided in a subsequent figure. One or more of hardware kernels K1 261a through K6 266a are also autonomous with respect to each other, thus allowing parallel processing of data, on a kernel-by-kernel basis, or
20 on a kernel-group by kernel-group basis. Because of this autonomy, and local control, the individual hardware kernels as well as the hardware kernel plane is data-rate scalable to a range of system clock rates.

For example, kernels K1 261a, K4 264a, and K5 265a are grouped together in hardware kernel group A 268a. Similarly, hardware kernel K3 263a is identified as a sole
25 kernel within hardware kernel group B 268b. These two exemplary kernel groupings provide a class of functions for the present host communication device which applies them to a wireless communication protocol application, as will be described in a subsequent flowchart figure. Hardware kernels, e.g., kernel K1 261a, are coupled to a configuration (or reconfiguration) bus 206a, e.g., via line 274. Configuration, status, and control information
30 are passed to kernels K1 261a through K6 266a via reconfiguration bus 206a, in the present embodiment. However, the present invention is well suited to passing different types of data and information using a wide variety of data lines and data bus configurations.

Reconfigurable interconnect 204a has an architecture that is primarily a reconfigurable logic 211, as described in Figure 2B. In this embodiment, reconfigurable interconnect 204a
35 provides connectivity between input/output lines of multiple kernels, or between input/output

lines of a kernel with components outside of kernel plane, e.g., allocator 219, processor 112 shown in Figure 2A, or other data buses (not shown). Data is passed between kernel plane 201a and the host communication device via an input/output line, e.g., line 122, that is coupled to reconfigurable interconnect 204a.

In one embodiment, reconfigurable interconnect 204a has only a limited amount of reconfigurability based upon the specific needs identified for a class of protocols in a given application, or for a class of applications. That is, based on an application, algorithm, function, operation, or class thereof, not all kernels will be required to have full interconnectability with all other kernels. Consequently, the present embodiment provides limited reconfigurability of interconnect 204a between kernels depending upon the needs of the application, function, algorithm, etc. for which a kernel is designed. The limitation on interconnectability provides the benefit of reconfigurability where it is needed, and restricts interconnectability where it is not needed. Thus, the inefficiency of a totally reconfigurable interconnect is tempered by identifying strategic scenarios where reconfigurability is appropriate. The strategic scenarios involve the class of functions to be performed, the design of individual kernels K1 261a through K6 266a to accommodate the class of functions, and the level of programmability provided for outside control. The common ground between the class of functions, operations, or algorithms is a case-by-case basis requiring analysis of variables such as mathematical basis, signal processing operations, algorithmic patterns, and silicon implementation.

Data is provided to and received from a kernel plane via data bus 122 or data line 130a. In the present embodiment, an input data line portion of data bus 122 is coupled to one side of reconfigurable interconnect 204a to provide data input to kernel plane 201a. Similarly, an output data line portion of data bus 122 is coupled to the other side of reconfigurable interconnect 204a to receive data from kernel plane 201a. Data that is provided to reconfigurable interconnect 204a is then routed to appropriate kernels K1 261a through K6 266a per configuration information provided to communication device. Alternatively, an input line portion of data bus 122 can be directly coupled to one or more of kernels K1 261a through K6 266a, e.g., if this functionality of a particular kernel is consistent across a range of applications. For example, if a kernel plane for a modem operation always initially performs an interpolation filter operation on input data regardless of the applications

5 within a class of communication protocols, then input data line may be routed directly to the kernel responsible for this function. The same coupling arrangement can be provided for data line 130a with respect to reconfigurable interconnect 204a and kernels K1 261a through K6 266a. While the present embodiment provides for less than full interconnectability, the present invention is well suited to providing full interconnectability between all kernels.

10 The modern signal processor is one instance of the heterogeneous reconfigurable architecture, which can be configured to provide a complete signal path for multichannel operation of a base-station. The hardware kernel processors were designed with a strong focus on applying the flexibility vs. computational efficiency trade-off to the specific needs of an application. As such, a rank ordering of the dominant computation-intensive kernels
15 found in the algorithms is identified. While the present invention provides an enumerated list of computational categories for a hardware kernel, the present invention is well suited to using specific quantities and types of categories as is appropriate for a given application. Bus 206a of Figure 2C is selectively reconfigurable to provide only the needed amount of interconnectivity to a kernel based upon the application, function, and/or algorithm, for which
20 a kernel is designed. For example, in one embodiment, kernel K3 263a does not require a status flag because the operation it performs requires no feedback and is run to completion. Thus, reconfiguration bus 206a provides no bus capability to kernel K3. In another embodiment, however, interconnectivity to provide communication of status information between a hardware kernel with another hardware kernel or allocator can be provided.
25 Referring now to Figure 2D, a block diagram of a kernel portion of a hardware kernel plane is shown, in accordance with one embodiment of the present invention. Kernel K1 261a provides one embodiment of many possible embodiments, which any of multiple hardware kernels in a kernel plane may use.

Kernel K1 261a includes a configuration information block 272 and a satellite kernel
30 block 270, coupled to each other by interconnect 276. Satellite kernel 270 has an input/output data line 278, which is a bus in the present embodiment, that provides communication with reconfigurable interconnect 204a of Figure 2C. Similarly, configuration information block 272 is coupled with reconfiguration bus 206a of Figure 2C, via configuration line 274. In one embodiment, configuration line 274 is a bus into configuration
35 information block 272, or can be a single line with multiplexed data. The amount of data the

5 bus or single line can handle can vary widely, depending upon the needs of an existing or projected application. Satellite kernel 270 is an electronic device, which is algorithmic specific in the present embodiment.

Configuration information block 272 is random access memory (RAM) in the present embodiment. However, the present invention is well suited to using any medium for
10 configuration information block 272 that can preserve and communicate a state condition for electronic devices. For example, configuration information block 272 can be registers, flash memory, or a state machine, e.g., using reconfigurable logic, that provides a bit stream of states to satellite kernel block 270. By having configuration information block 272 as a local dedicated source, that can also be controlled local to satellite kernel 270. This arrangement
15 provides a very quick and efficient changing of configuration data for algorithmic satellite kernel 270. Consequently, time-sharing of a hardware kernel is feasible and practical in the present embodiment.

In the present embodiment, hardware kernels e.g., K1 261a through K6 266a of Figure 2C, have been designed to fit into one of multiple categories of data processing applicable to
20 wireless communication. The category of data processing refers to the operational speed of the hardware kernel, which includes an energy-flexibility tradeoff. The specific category for which a hardware kernel is designed is determined from the number and type of operations per sample of data processed in the hardware kernels.

The kernel processors cover the multi-standard TDMA signal processing requirements, and can be categorized corresponding to classes of MOPS. In the present
25 embodiment, signal processing for a wireless communication application includes the following classes of MOPS: 1) Code Demodulation/Dechannelization; 2) Code Generation; 3) Parameter Estimation; 4) Sequence Alignment and Combining; 5) Equalization (optional); and 6) Front-end Processing.

30 Satellite kernel 270 includes a controller 271 and a configurable computation kernel (or algorithmic-specific computing element) 273a, coupled to each other via a clock line 279 and a control line 284. Configurable computation kernel 273a is also referred to as a computing element or a processing engine.

Controller 271 includes a state machine with memory, in the present embodiment, that
35 is capable of controlling configurable computing element 273a. In another embodiment,

5 controller 271 includes only memory that is capable of preserving state conditions of at least one configuration of configurable computing kernel 273a. To achieve distributed control, kernel K1 261 is equipped with an interface that enables it to exchange data streams with other kernels efficiently, without the help of a global controller. Hardware kernel K1 261a uses a distributed control and configuration via local controller 271, which effectively reduces
10 overhead in terms of instruction fetch and global control. Kernel K1 261a also includes an interface, e.g., in configurable computation kernel 273a, that enables it to exchange data streams, e.g., data line 278, with other kernels efficiently, without the help of a global controller. The communication mechanism between each kernel is dataflow driven in the present embodiment. Local controller 271 can provide local control signals for initiation,
15 reset, and interrupt for configurable computation kernel 273a, as well as scaled clock rates.

In the present embodiment, configurable computation kernel 273a is designed specifically to perform a given algorithm, function, operation, or class thereof. Therefore, satellite kernel 270 has flexibility, e.g., reconfigurability, within the class of functions, operations, or algorithms to which it has been designed. By virtue of the fact that
20 configurable computation kernel 273a is designed for a relatively narrow application in the present embodiment, it is consequently very energy efficient. Thus, it meets the needs of a wide range of communication protocols within a spread spectrum category, while being very efficient. Additionally, because satellite kernel 270 has its own local controller 271, it operates autonomously with respect to the balance of the kernels in a hardware kernel plane, and to the balance of the communication device. Thus, satellite kernel 270 can be activated or bypassed for a given function of an application, depending on the needs and protocol
25 chosen for the application. A description of the configuration and operation of a satellite kernel 270 is presented in a subsequent flowchart. The present architecture is well suited to a wide range of data processing functions, operations, and applications besides communication
30 applications.

In the present embodiment, computing element 273a includes an architecture of electronic devices with coupling arrangements, from one or more of the possible techniques described in Figure 2B. That is, depending on the function, algorithm, operation, or class thereof, being implemented by the hardware kernel, computing element 270 can include any
35 combination of the techniques for device choice and configuration, including reconfigurable

5 logic 211, reconfigurable datapath 221, reconfigurable dataflow 231, or reconfigurable logic
241. In the present embodiment, the computing element in a hardware kernel, e.g.,
computing element 273a of KI 261a, can exploit any combination of the four types of
reconfigurability, in an architecture referred to as Dynamically Reconfigurable Logic (DRL).
However, the present invention is well suited to incorporating other types of computational
10 models that have different levels of granularity or different applications of granularity.
Additionally, the techniques of Figure 2B used in configurable hardware kernel can be chosen
depending upon the uncertainty of a design or function within the communication device.
Thus, by providing a very flexible architecture to an autonomously controlled configurable
hardware kernel for the narrow scope of an uncertain function or algorithmic technique, the
15 present invention frugally allocates the most flexible reconfiguration resources. However, the
present invention is well suited to complementing these enumerated techniques with other
configuration and architecture techniques.

Because the computing element 273a is function (or algorithmic) specific, each of the
techniques used is subsequently function specific. Thus, the electronic devices and their
interconnections can utilize function-specific reconfigurable logic 211, function specific
reconfigurable datapath 221, function-specific reconfigurable dataflow 231 and/or function
specific reconfigurable logic 241 techniques as shown in Figure 2B in one embodiment. The
function-specific aspect of the devices and their interconnects means that the device is only
effective and useful for the intended function, sub function, or classes thereof, in this
embodiment. By doing so, this architecture efficiently delivers a class of MOPS with
25 flexibility in the configuration of these MOPS and scalability across data rates and channel
densities.

Electronic devices refer to the basic building blocks of electronic circuits such as
transistors, diodes, resistors, conductors, and other elements that are well known in the art.
30 The collection of electronic devices and interconnects can be figuratively divided into a fixed
grouping 275a and a flexible grouping 275b, intercoupled to each other on a device level, as
required by the function implemented therein. For example, in one embodiment, flexible
architecture can be used to selectively group and couple registers to implement a class of
functions whose math operations vary by their bit length, depending on the protocol used.
35 Thus, each of the multiple hardware kernels described in Figure 2C have an architecture that

is tuned to its intended function. In the present embodiment, the combination of architecture in a computing element is dependant upon the functions, or class of functions, to be performed by the hardware kernel. Other variables, such as performance requirements, user preferences, future expandability into undefined protocols are also included as inputs in the choice of architectures. Because the hardware kernels each have a discrete function, operation, or class thereof, they can be evaluated as true object-oriented hardware.

Thus, a channel element can be built-up from the set of configurable hardware kernels to realize a reconfigurable multi-channel digital base band modem signal path that performs all the digital modulation-demodulation as well as channel encoding-decoding required per logical channel for all narrowband and wideband telecommunication standards. In the present embodiment, kernel plane can form a modem card in a systematic and modular fashion in modules of multiple channels per card, depending on their radio (cell-site) system planning. The present invention can be adapted to accommodate a wide range of channels.

In the present embodiment, two or more types of configurable architecture techniques are used in a given hardware kernel. However, the present invention is well suited to using a single type of configurable architecture is used in a given hardware kernel. Additionally, the kernel compositions can vary within a hardware kernel plane, and between hardware kernel planes. Multiple types of architecture can be strategically located and coupled within a hardware kernel to accommodate the particular variation in the function/sub function desired. For example, if the variation for sample select sub function over 3GPP, 3GPP-FDD, 3GPP-TDD, and IXtreme protocols includes the number of bits selected, then the hardware kernel includes a reconfigurable logic for the interconnect bus and the storage location associated with the range of bits and a reconfigurable datapath for the balance of the circuit.

The present invention is well suited to using a wide range of architectural techniques shown in Figure 2B, and combinations thereof, from which individual hardware kernels are designed, constructed, and operated. These hardware kernels are capable of performing a wide range of functions within a class that spans a wide range of applications. Exemplary functions, for which kernels can be configured, are shown in Figures 2B, 2C, and 5.

Several exemplary hardware kernels have been defined in related co-pending patent applications and are applicable in the present communication device, e.g., 100 of Figure 1b.

While these related patent applications provide a specific function for hardware kernels, the

present invention is well suited to a wide range of data processing functions for electronic devices. These commonly assigned and related applications, which are incorporated herein by reference, include:

- 1) "A CONFIGURABLE CODE GENERATOR SYSTEM FOR SPREAD SPECTRUM APPLICATIONS," U.S. patent application serial number 09/751,782, filed on December 29, 2000;
- 2) "A CONFIGURABLE MULTIMODE DESPREADER FOR SPREAD SPECTRUM APPLICATIONS," U.S. patent application serial number 09/751,785, filed on December 29, 2000;
- 3) "A CONFIGURABLE ALL-DIGITAL COHERENT DEMODULATOR SYSTEM FOR SPREAD SPECTRUM APPLICATIONS," U.S. patent application serial number 09/751,783, filed on December 29, 2000; and
- 4) "METHOD AND APPARATUS FOR PROCESSING A SECONDARY SYNCHRONIZATION CHANNEL IN A SPREAD SPECTRUM SYSTEM," U.S. patent application serial number 09/772,583, filed on January 29, 2001.

The term architecture describes the electronic devices and coupling arrangements used in configurable hardware kernel plane 201a of Figure 2C, Kernel K1 261a and configurable computation kernel 273a of Figure 2D, reconfigurable interconnect 204a of Figure 2C, and the specific exemplary hardware kernels provided in the aforementioned applications. The coupling arrangements include interconnect routing, grouping, and hierarchy. The various combinations of reconfiguration techniques 211, 221, 231 and 241 of Figure 2B also describe the architecture of the configurable computation kernel 273a, the reconfigurable interconnect 204a, and the specific exemplary hardware kernels. Devices can include components such as gates, selective interconnects, memory, lines, buses, and a wide range of conventional devices that are chosen and coupled in order to satisfy the functional requirements of a given application. More information on architecture of configurable devices can be found in the text "Software Radio Architecture," by Joseph Mitola III, which is hereby incorporated by reference.

Referring now to Figure 2E, a block diagram of a hardware kernel containing a subcomponent hardware kernel is shown, accordance with one embodiment of the present invention. Hierarchical block diagram 120b has many components and coupling

5 arrangements that are similar to those presented in Figure 2D. For purposes of clarity, only a description of components, coupling arrangements, and alternative embodiments for Figure 2E that are different from Figure 2D will be provided herein; otherwise, the description of components, coupling arrangements and alternatives provided in Figure 20 will apply similarly to the present figure.

10 Satellite kernel 270a includes a hierarchy of hardware kernels, wherein another hardware kernel K7 317a is included therein. Kernel K7 317a, having an exemplary configuration as K1 261a of Figure 2D, provides a dedicated support kernel function to its parent kernel K1 311a. An additional control line 284a and clock line 279a is provided to dedicated hardware kernel K7 317a. Due to the hierarchy, dedicated hardware kernel K7 317a
15 can step up a clock rate another level for its own functions if needed for a given application.

20 Different configurations of a hardware kernel within a hardware kernel can exist depending on the needs for a given function in a given application. For example, an exemplary cellular telephony function can tailor parent kernel K1 311a for performing a demodulation operation or a matched filter operation. Either of these functions can benefit from a dedicated hardware kernel, e.g., K7 317a, providing dedicated code generator functions. In this manner, hardware kernel K7 317a can run autonomously under the control of its parent kernel K1 311a. This configuration and architecture provides local control and operation, which consequently reduces traffic on a system processor or controller. The present invention is well suited to a variety of hierarchical levels or parent hardware kernels
25 having different quantities of dedicated kernels therein.

Referring now to Figure 2F, a block diagram 200f of the inputs, outputs, and functions accommodated by a hardware kernel in an electronic communication device is shown, in accordance with one embodiment of the present invention. Functions described in block diagram 200f represent operation of exemplary hardware kernel(s) of Figures 2C through 2F.

30 Handshake input 280, input flags 282, and configuration information 281 is received at control signal generation block 284. Handshake input 280 is a signal from a source outside of a hardware kernel that initiates the operation of the hardware kernel, via a control signal generation block 284. Input flags 282 alter the data, states, or functions of a hardware kernel. For example, a handshake input 280 can enable the hardware kernel functions (or algorithmic
35 computations) 292 to proceed, while an input flag 282 will enable an appropriate

5 communication of a previous state condition for hardware kernel functions 292. Note that the specific conditions, uses, and quantity of input flags can vary widely according to the needs of a particular function in a particular application, and the capacity of the hardware kernel functions. Configuration input 281 provides information on how to configure hardware in a hardware kernel so as to perform the desired algorithm computation function. Configuration
10 input 281 includes information specifying: 1) type of operations to be performed; 2) inputs; 3) outputs; and 4) parameters of the configurable hardware kernel. Configuration input 281 can also include information regarding the interconnect configuration used to build a cluster, or group, of kernels for implementing a function or sub function.

Control signal generation block 284 includes sub functions of preserving e.g.,
15 recording, and transmitting a state or a configuration of a hardware kernel as directed by handshake input 280 and configuration input 281. Control signal generation block 284 also includes a sub function of clock scaling. Lastly, control signal generation block 284 generates output flags 296 and a handshake output 298 to components outside of the hardware kernel. Thus, in the present embodiment, control signal generation block 284
20 provides a control signal, via a bus or a multiplexed line, to algorithm computation function block 292 that includes an enabling/status signal 295c, a state condition signal 295b, and configuration information signal 295d. Control signal generation block 284 also provides an adapted clock signal 295a to algorithm computation block 292.

Clock scaling sub function can speed up, slow down, or preserve the rate of the
25 system clock signal 288 received at control signal generation block 284. The specific rate of the adapted clock signal 295a depends upon the rate of data processing for which algorithm computation block 292 is designed. Clock signal input 288 is a system clock of the communication device in the present embodiment.

In one embodiment, handshake output 298 and output flags 296 indicate the status of
30 a hardware kernel or of data processed by algorithmic computation block 292. For example handshake output 298 or output flags 296 include information such as successful execution, error rate, state condition, etc.

Algorithmic computation block 292 performs a desired algorithm for a set of data,
received as input data 290. Inputs 295b, 295c, and 295d from control signal generation block
35 284 allows algorithmic computation block 292 to essentially perform its algorithmic-specific

function autonomously from the balance of the communication device. During or after the completion of its function, algorithmic computation block 292 provides output data 294 to sources outside of hardware block, and provides status information 295 to control signal generation block 284 for subsequent handshake or flag setting.

Algorithmic computation block can be figuratively divided into a fixed portion of the algorithm 292a and a flexible portion of the algorithm 292b, communicating information to each other. The fixed portion block 292a of the algorithm computation is the portion whose algorithmic functions essentially do not change for different communication protocols, e.g., the core functions or the common math that can be extracted from the different algorithms required for each protocol. In contrast, the flexible portion 292b of the algorithm computation is the portion that does change for different communication protocols. In one embodiment, the fixed portion 292a and flexible portion 292b of the algorithmic computation corresponds to the fixed portion 275a and flexible portion 275b of configurable computation kernel 273a shown in Figure 2D.

In the present embodiment, function blocks in block diagram 200f are implemented by hardware components described in Figures 2A through 2F. For example, one embodiment implements algorithmic computation function 292 of Figure 2F via computing element 273a of Figure 2D, and implements control signal generation block 284 in controller 271 and configuration information block 272. Clock signal input 288, handshake input 280, and input flags 282 are provided to control signal generation block 284 via configuration line 274, and are communicated between components in hardware kernel 261a via an interconnect line 276, clock line 279, and control line 284. Handshake output 298 and output flags 296 are transmitted on configuration line 274. Input and output flags, and handshake input and output, can be provided to/from another hardware kernel in the hardware kernel plane or to/from components outside of hardware plane 201a, via reconfiguration bus 206a. While the present embodiment provides specific hardware components and configurations for implementing hardware kernel functions 200f, the present invention is well suited to using a wide variety of hardware components and configurations for accommodating locally controlled algorithmic specific function blocks.

The present embodiment of components, configuration and functionality of hardware kernels shown in Figures 2B through 2F are capable of providing an operating efficiency

from greater than 10 to several hundred (100) MOPS/mW in the present embodiment. This rate is well ahead of what is achievable from a programmable digital signal processor. Yet the present embodiment also provides a level of flexibility over a wide range of functions and a level of user control that is well beyond that of a traditional ASIC. The present invention is well suited to deliver a wide range of computation power and computational efficiency, depending upon the semiconductor process technology, and VLSI integration/physical design method.

Referring now to Figure 3, a block diagram of a modulation/demodulation (modem) function block of an electronic TDMA communication device is shown, in accordance with one embodiment of the present invention. Modem function block 320 includes a data switching demultiplexer 329 and multiple functional planes, e.g., functional kernel planes [1] 330a and [i] 330i that represent the functions required and performed by kernels for a given application. Receive path 354 shows the direction of data flow through the functional blocks 301 through 318 in a modem function plane 330a. In the present embodiment, the functions provided are for an exemplary cellular application. Each functional plane represents the capabilities of a configurable demodulating hardware kernel plane. While the present embodiment shows only two functional planes, the present invention is well suited to using any number of functional planes, as appropriate for a given application. In the present embodiment, functional planes [1] 330a through [I] 330i perform modem functions for any one of a multitude of different communication protocols, e.g., IS-136, GSM, as well as anticipated future protocols. While the functions listed in the present embodiment are focused towards TDMA, similar techniques are well suited to implementing functions that are applicable to code division multiple access (CDMA) protocol, as described in co-pending application for "A WIRELESS SPREAD SPECTRUM COMMUNICATION PLATFORM USING DYNAMICALLY RECONFIGURABLE LOGIC," U.S. patent application serial number 09/772,584, filed January 29, 2001. The specific communication protocol implemented in a functional plane depends upon how the functional plane was configured.

Each functional kernel plane provides a novel programmable dataflow architecture for a complete receiver for TDMA systems. The architecture can be used to detect signals in any TDMA-based multiple-access communication system. The architecture comprises a signal detection path and a parameter estimation path. The dataflow architecture consists of a

5 sequence of programmable arithmetic kernels optimized for specific functions that are found in TDMA modems.

The signal processing path includes an interpolating filter 301, a pulse shaping matched filter 302, a sample selector 303, a derotator 304, a scaler 305, a channel matched filter 306, an equalizer 307, a decrypter 308, a deinterleaver 309, a channel decoder 310 and a
10 block decoder 311, connected in cascade. The receiver also includes parameter estimator circuitry including a received signal strength indicator 312, an automatic gain control circuit 313, a frequency offset estimator 314, a channel impulse response (CIR) estimator 315, a CIR updater 316, a delay-epoch estimator 317 and a received signal quality indicator 318.

The received digital signal is a complex (in-phase and quadrature components) signal
15 that travels through the detection path, which commences at the input to the interpolation filter 301. This filter is a digital filter whose purpose is to up-sample the incoming signal to a rate commensurate with the processing in the matched filter and sample select functions that follow. This rate is based on the sample-timing accuracy required for the detection path to operate to meet a minimum performance criteria. The characteristics of this digital filter (taps, coefficients) can be programmed by the user to meet the requirements. The received signal is then passed through a pulse-shaping matched filter 302, which serves the purpose of a Nyquist filter. Depending on filter order and coefficients, filters 301 and 302 may be combined for greater efficiency.

The signal is then passed through a sampler 303 that selects the data sample
20 corresponding to the phase of a timing correction signal coming from the delay-epoch estimator 317.

The signal is then passed through a derotator 304 that performs a complex multiplication to remove the frequency offset typically present in the incoming signal. Correct demodulation operation cannot occur until this has been performed.

30 The data is then adjusted in the complex plane by a scaler 305 using an IQ scaling function that adjusts the amplitudes of the inphase and quadrature components based on a correction signal from the RSSI and AGC estimation blocks 312, 313, respectively.

The data is then passed through a channel matched filter 306 that a filter whose coefficients are matched to the response of the transmission channel. Since the transmission
35 channel is changing in a mobile communication system, a channel estimation process is

5 required to continually update the coefficients of this filter. This estimation operation is provided by the channel impulse response (CIR) estimator 315.

The output of the channel matched filter is passed to equalizer 307 that is realized via a least-means squared equalization with or without decision feedback, or a maximum-likelihood sequence estimator, depending on performance requirements. The equalizers in this dataflow mode can be operated in both forward and time-reversed mode, as well as in a standard or reduced-state sequence estimation mode. The data out of the equalization process is a stream of soft-decision output symbols.

These symbols are then fed into a decryption block 308, if required. Some TDMA transmission standards perform a simple encryption of the inner receiver link, and these encryption methods typically involve simple hashing functions. Then the decrypted symbols are supplied to a deinterleaver block 309 to undo the coded-symbol interleaving operation performed at the transmitter.

The data from the deinterleaver is read into a channel decoder 310 to perform the channel decoding operation, which is implemented via a convolutional decoder. The quality of the received signal is measured via an received signal quality metric indicator 318 using an signal-energy-to-noise estimation algorithm, and this quality metric is used to condition the signal at the output of the channel decoder. In some systems, a concatenated coding scheme is used to improve error protection and achieve even higher link performance as measured by lower bit-error rates. In such a case, the data from the convolutional coder is fed into a block decoder 311 to correct the block errors present in transmission over fading channels.

As will be apparent, the signal path through the detection path is only feed forward, and it lends itself very well to a highly pipelined implementation of this dataflow architecture.

The signal estimation path consists of a series of parameter estimation kernels connected to the detection path as shown in Figure 3. The basic method of operation of any parameter estimator is to capture an observation (signal) and compute an estimate of an unknown parameter required to complete the accurate detection of the transmitted signal (which is also an unknown to the receiver). Typically, a known signal often called a pilot or training sequence, is embedded in the TDMA transmission so that the estimation path can then operate on estimating the unknown transmission parameters using these known signals.

There are several unknown parameters that the TDMA receiver must estimate in order

to operate. First, the timing parameter estimation operation is used to estimate the correct sampling phase of the received digital signal. The CIR estimator output 315 is passed to a delay-epoch estimator 317 that uses a loop-filter to track the timing drift of the samples from their ideal position and deliver a correction signal to sample the data from the sample select block 303.

The frequency offset estimation operation is performed by the frequency offset estimator 314 via a quadricorrelator or maximum-likelihood frequency estimation algorithm. The frequency offset estimate is applied to the derotator 304 in the detection path to correct the error in the incoming signal.

The channel impulse response is estimated via CIR estimator 315 which comprises a maximum-likelihood estimator of a random time-varying fading multipath channel. This estimator often takes the form of a Kalman filter, a multislot averaging filter, and interpolation filtering. The CIR estimate is then updated using the CIR update function 316 which is implemented via a averaging filter which updates the CIR coefficients used in the equalizer 307 in the detection path.

The gain control settings required to adjust the amplitude of the signal at both the input to the A/D converter and the input to the channel matched filter in the detection path are obtained via the combination of received signal strength indicator (RSSI) estimator 312 and automatic gain control circuit (AGC) 313. The RSSI estimator 312 is usually implemented via a time-averaging filter and energy squaring operation, and AGC 313 is implemented via a first order feedback loop that generates a correction signal that maintains the input signal within an amplitude range.

All of the parameter estimation kernels shown fit into a dataflow architecture where their feedback signals are updated on a symbol, slot, or frame basis, which allows for an efficient pipelined implementation since the feedback loops are closed typically on the orders of kilohertz.

In the present embodiment, additional functional planes, e.g., functional plane [2] 330b through plane [i] 330i, include the same functions as those performed by functional plane[1] 330a. Thus, every functional plane in modem function block 320 is equally configurable with the full flexibility to accommodate each of the communication protocols.

In another embodiment, one or more functional planes have functional capabilities that are

different from each other. For example, in one embodiment, each functional plane in modem function block 320 has the ability to handle different subsets of the superset of communication protocols accommodated by the overall communication device. Thus, one functional plane might be configured to accommodate 3GPP communication protocols, while another functional plane is configured to accommodate a legacy communication protocol.

Because a functional plane can accommodate multiple communication protocols, it is configurable to a desired protocol. On a larger scale, each of the multiple functional planes can be configured to accommodate the same communication protocols at a given time, or to accommodate different communication-protocols at a given time. Thus, for example, functional planes [1] 330a and [i] 330i can be configured to both perform a GSM protocol at a given point in time. Alternatively, functional planes [1] 330a and [i] 330i can be configured to perform different communication protocols at a given point in time, e.g., functional plane [1] 330a is configured to perform a 3GPP protocol while functional plane [i] 330i is configured for a legacy communication protocol.

One aspect of the configurability of a functional plane of Figure 3, is that the configuration itself can change, e.g., it can be reconfigured. The independent variable associated with the change can be time, a channel designation, or some other user-defined variable. The time variable can be a long or short temporal designation, e.g., years, months, or even milliseconds. Furthermore, the reconfigurability of the functional planes is dynamic in one embodiment, e.g., the reconfiguration occurs while other functional planes are operating. The configuration data used to designate the functions and protocol accommodated may be communicated via wireless transmission of configuration data along with the data, or by loading pre-stored configuration data on communication device. In this manner, the present invention provides a temporal and functional range of granularity for the functional planes of the communication device that are definable by a user.

In the present embodiment, multiple functional planes, e.g., 330a through 330i, are physically implemented in a single configurable modem processor block, e.g., block 102a of Figure 1B. However, the present invention is well suited to a wide variety of physical implementations via a wide range of methods.

In particular, a channel element can be built-up by the user from the set of reconfigurable kernels to realize a reconfigurable multi-channel TDMA digital base band

modem signal path that performs all the digital modulation-demodulation as well as channel encoding-decoding required per logical channel for all narrowband and wideband TDMA standards. Some possible configurations that a user can realize by configuring the hardware kernels and the configurable interconnect in the communication device are summarized below, for the forward (downlink) and reverse (uplink) links:

GSM Configurations:

- Standard 8-slot (to be specified)

HCS D Configurations:

- Time-slots from standard GSM configured to data mode

GSM-GPRS Configurations:

- Initial Deployment:

- Downlink: 1 slot in GPRS mode
- Uplink: 1 slot in GPRS mode

Future:

- Downlink: multi slot in GPRS
- Uplink: multi slot in GPRS

IS-136 Configurations:

- Voice: Standard Configuration (to be specified) 3 time slots per carrier
- Data: Single slot or multislot configured into data mode

IS-136+

- Initial Deployment:

- EDGE-Compact (1/3 Reuse factor for deployment, based on total spectrum available)

Future:

- EDGE-Classic (4/12 Reuse factor for deployment...)

Modem signal processor function block 320 offers a communication system, allowing a very high level of channel integration, while allowing the communication system to employ proprietary algorithms to improve radio link performance, as will be described hereinafter. Each channel element, including its definition and structure, is completely under the communication system's control using the mixed programming granularity to control

5 dataflow, controlflow, and interconnect. This process is discussed in more detail for a subsequent flowchart figure.

Transmit path 352 is configured with multiple channel elements in the present embodiment, thus enabling maximal use of forward-link capacity per sector in multicamer systems. The architecture of the modem signal processor 102a of Figure 1B is such that it
10 enables the system designer to program the chip at many levels, including control of specific datapaths to realize different algorithms. A hierarchical Application Programming Interface (API) that supports several levels of user control for the modem signal processor is described in co-pending patent application entitled "A METHOD OF GENERATING A
CONFIGURATION FOR A CONFIGURABLE SPREAD SPECTRUM
15 COMMUNICATION DEVICE, Serial No. 09/772,582, filed January 29, 2001.

Referring now to Figure 4, a block diagram of encode/decode (codec) functions accommodated by the electronic communication device is shown, in accordance with one embodiment of the present invention. Codec function block 400 includes an address generator block 401 coupled to an allocator function block 402, in turn coupled to dynamically
20 assignable scratch/buffer memory 404, such as random access memory (RAM), in the present embodiment. Dynamically assignable scratch/buffer memory 404 is coupled to each of the multiple possible configurable decoder functional planes. Allocator function block is implemented by allocator hardware block 219 of Figure 2A, which includes state machine components and memory that are chosen and coupled in a manner to manage multiple
25 functional planes, e.g., planes 406 and 410. The configuration of the allocator components can vary depending upon the protocol implemented in the multiple functional planes, e.g., planes 406 and 410. Dynamically, a single scratch/buffer memory 404 is operational to provide configuration and state information as required for configuring and sharing resources in the multiple functional planes 406 and 410.

30 The present embodiment of codec function block 400 includes a single functional kernel plane for a given data flow direction, e.g., decode functional plane 410 for decode data path 408, and encode function plane 406 for encode data path 409. Each functional plane utilizes configurable codec hardware kernels to accommodate their different functions, which are described hereinafter.

35 As an illustration, decode functional plane 410 includes an exemplary arrangement of

sub functions for a decode path 408 to translate encoded received data into a data signal, per an appropriate communication protocol for the desired application. The arrangement of functions includes, in one embodiment, a bit field extraction block 410 coupled to a memory block 412, which is then coupled to deinterleaver block 414, which is in turn coupled to rate matching block 416. Rate matching block 416 is coupled in parallel to Viterbi block 418 and to Turbo decoder block 420, both of which are then coupled to provide data to cyclic redundancy checker (CRC) block 422. Lastly, CRC block 422 is coupled to buffer 424. A similar, complementally coupled arrangement of sub-functions exists on encode functional kernel plane 406, but is omitted for clarity.

The function blocks provided in Figure 4 can be implemented on corresponding individual hardware kernels, e.g., kernel K1 261a through K6 266a. However, some functions may share a common hardware kernel if the types of operations (e.g., math operations) and the processing rate (e.g., symbol rate) are similar enough and if scheduling of the hardware kernel and the data flow through the hardware kernel are satisfactory.

Decode function kernel plane 410 includes other user-configurable decoder functions, described hereinafter, for convolution decoding and turbo decoding. The basic function of convolution decoding and turbo decoding is known by those skilled in the art. The convolution decoder function has the following user-programmable (or user-configurable) parameters: code polynomial; code (R, K), with $K=5-9$, data rate; blind rate-detection for voice channels; rate matching. Convolution decoder function also has user-programmable depuncturing pattern, traceback method, and soft-decision output. Turbo decoder function has user-programmable: code polynomials ($K=3,4$); data rate; block size (up to 6120); number of iterations; termination conditions; decoding metric (max-log-MAP, user-specified correction table); input scaling; traceback method; and depuncturing pattern.

In contrast, encode functional plane 406 includes a sequential arrangement of functions, or sub functions, (not shown for clarity) for an encode path 409 to translate data into an encoded signal, per an appropriate communication protocol of the desired application. Encode path 409 shows a data path direction through codec function block 400 that is opposite that of decode path 408. In this manner, the codec function block 400 can accommodate both directions of data traffic, by time-sharing the functional resources. A wide range of common functions, implemented serially, and a wide range of diverse

functions, implemented in parallel, for the wide range of TDMA applications can be obtained from the operating specifications for these different protocols. In particular, transmit encoder path 409 of Figure 4 includes user-configurable encoder kernels for convolution encoding and turbo encoding. The following functions make up the encoding kernels: Convolution Encoder function and turbo encoder function; convolution encoder includes the following user-programmable parameters: code polynomial (R, K), with $K=5-9$; rate matching; puncturing pattern. Turbo encoder includes user-programmable: code polynomials ($K=3,4$); data rate; block size; rate matching; puncturing pattern. The specific function block descriptions implemented by reconfigurable decoder plane 406 include deinterleaver controller, turbo decoder, and convolution decoder.

The channel codec function plane 400, as implemented in channel codec signal processor 104 of Figure 1B, operates under the following criteria, in one embodiment:

- A maximum of 32 logical channels are available per carrier per configuration
- A configuration, as defined as a radio channel configuration, maps directly onto one thread of processing on the channel codec signal processor
- Type of channel specified by BTS channel card controller.
- Multiple radio channel configurations must be supported across a multiple standards, including all radio configuration characteristics for the reverse channel in derivative systems.
- BTS channel card controller will interface to the BTS cell controller for sending and receiving control information associated with call setup, teardown, and handoff.
- The BTS cell controller will not demand a grand total of channel bandwidth beyond the capacity of the channel codec signal processor(s). If this cannot be guaranteed, then a protocol on dropping channels beyond the capacity limit must be specified.
- Assignments for radio configurations for specific channels will be made prior to the arrival of the frame; i.e. assignments for FRAME (N) will be sent prior to the arrival of FRAME (N).
- The logical channel assignments for a physical channel (for the BTS) can be changed during the course of a call at any FRAME boundary.
- Physical channels may request a larger bandwidth or smaller bandwidth for the same assignment during the course of a call.
- Logical channel assignments will be maintained so there are no holes in the

5 assignment table. This provides the capability for adding the largest (widest) new channel assignment within the remaining capacity.

- Channel assignments and deassignments require one time slot (only one assignment or deassignment per time slot). This assumes cleanup occurs upon every assignment/deassignment.

10 • Resource consolidation/defragmentation will be performed immediately after a deassignment or assignment.

While the present embodiment shows only two functional planes, the present invention is well suited to using any number of functional planes, as appropriate for a given application. In the present embodiment, functional planes 406 and 410 can be configured to perform codec functions for a wide range of communication applications, as described hereinabove. For example, in one embodiment, multiple functional planes (not shown) for both encoding and/or decoding can exist. In the present embodiment, additional functional planes for a given data flow, e.g., decoding path 408, include the same functions for decoding. Thus, every functional plane in codec functions is equally configurable with the full flexibility to accommodate each of the communication protocols. In another embodiment, one or more functional planes have functional capabilities that are different from each other. For example, in one embodiment, each functional plane in codec function block 400 has the ability to handle different subsets of the superset of communication protocols accommodated by the overall communication device. Thus, one functional plane can be configured to accommodate GSM communication protocols, while another functional plane is configured to accommodate IS-136. All the functional planes can be physically implemented in a single codec hardware processor block, e.g., block 104 of Figure 1B, in the present embodiment. However, the present invention is well suited to a wide variety of physical implementations. The configurability and dynamic reconfigurability of codec functional planes is also described in subsequent flowcharts.

Referring now to Figure 5, a diagram is shown of the separating and combining process of a single thread into multiple concurrent threads to be accommodated on the communication device, in accordance with one embodiment of the present invention.

The model for computation using the reconfigurable multiprocessor architecture of the present invention starts from the model of a single thread representing function A 504, which

5 is initialized on a microprocessor or digital signal processor, e.g., processor 112 of Figure 1B. Several application-specific processing threads, e.g., thread 1 501 through thread 3 503, are split off in the pre-processing stage. For example, in one embodiment, function A 504 is a modem function within a given application, e.g., a wireless communication protocol such as TDMA IS-136. Function A 504 includes multiple sub functions that can be performed in
10 series or in parallel.

The sub functions that are performed concurrently, e.g., in parallel, are referred to as threads, e.g., thread 1 501, thread 2 502, and thread 3 503, for the modem function A 504. The number of threads can vary in different embodiments, according to the operations required by a given function or sub function, and their need to be performed concurrently,
15 e.g., defined by a communication protocol in the present embodiment. Concurrent operations 508 are performed on each of the respective threads 501-403 in a communication device, according to the sub function requirements. Thereafter, the threads are terminated with the provision of data and control is returned the microprocessor. Each thread can be implemented on an autonomous configurable kernel in the present embodiment. A processor
20 112 along with an optional allocator 219, as shown in Figure 2A, can initiate or manage the separating and combining operations.

The chronological sequence of events for the function of separating and combining functions/sub functions is provided in a subsequent flowchart figure. In one embodiment, a modulation function can be broken down into concurrent sub function on data, such as
25 interpolation filter, sample select, etc. Similarly, each sub functions can be broken down into multiple concurrent operations. For example, a filter interpolation requires operations such as data fetch, addition, decision logic, etc.

FLOWCHART IMPLEMENTATION OF PROCESSES

30 Referring now to Figure 6A, a flowchart 6100 of the process used to implement a design configuration onto a configurable electronic communication device is shown, in accordance with one embodiment of the present invention. By using the flowchart embodiment of the present invention, a configurable electronic device can be configured to a user-specific design configuration. In this manner a significant amount of control over the
35 operation and function of the configurable device is provided to the user. The

5 implementation of the design into the device is direct and convenient, thus providing timely flexibility in a field where protocols can change quickly and frequently. Flowchart 6100 is implemented, in general, by communication device 100 of Figure 1B and VMI 150 of Figure 1D, as well as diagrams of components as shown in Figures 2A through 2F.

Flowchart 6100 begins with step 6102. In step 6102 of the present embodiment, a
10 design configuration is received at a configurable device. For example, a design configuration for a configurable communication device can be a 3GPP-specific configuration, implemented by user-specified proprietary algorithms, in one embodiment. Additional description on the process for designing a configuration for a configurable device is provided in co-pending patent application entitled "A METHOD FOR DESIGNING A
15 CONFIGURATION FOR A CONFIGURABLE SPREAD SPECTRUM COMMUNICATION DEVICE." Serial No. 09/772,582, filed January 29, 2001.

Step 6102 is implemented, in one embodiment, by providing design configuration data, e.g., from configuration mappings input 174 of Figure 1D, to configuration information block 272 of a hardware kernel K1 261 a via reconfiguration bus 206a. More than one
20 configuration can be provided in one embodiment. In this case, a configurable device can be partitioned to perform multiple protocols, having a unique configuration for one or more functions, within a single communication device. The partitioning can be static amongst the hardware, or can be dynamic, e.g., in the case of time-shared configurable hardware kernels. The configuration information can also contain information to provide options, e.g., quality of
25 service (QOS) when implementing the function for a given user or channel in the communication device. The transmission of the design configuration can either be via wired or wireless coupling.

While the present embodiment for step 6102 provides a specific location of configuration information block 272 and a specific route, e.g., reconfiguration bus 206a, in
30 which it is downloaded to a hardware kernel, the present invention is well suited to using alternative components and interconnects to implement step 6102. By providing the configuration information at a level that is local, e.g., in terms of spatial and control aspects, to the individual hardware kernel, the present embodiment provides effective local processing that is autonomous, in varying degrees, to the balance of the host device. In this manner, the
35 present invention is able to provide efficient and flexible parallel processing of data.

5 Step 6102 can be implemented by receiving desired configurations for hardware kernels, e.g.,
kernel 261a of Figures 2C through 2F, in communication device 100 of Figure 1B, via a
variety of embodiments. For example, in one embodiment, configuration information is
received via wired communications with a computing device, e.g., a workstation. In another
embodiment, configuration information can be provided by an electronic storage medium,
10 e.g., CD-ROM. In yet another embodiment, configuration information is received by wireless
transmission from another communication device via antenna 120. Furthermore,
configuration information is provided at the time communication device 100 is manufactured
and/or initially programmed for operation in the field, in the present embodiment. However,
in another embodiment, configuration information is dynamically implemented at a time
15 communication device 100 is in operation in the field. Configuration information is received,
processed, and implemented via system processor 112 (or BTS card controller 110a) and
system memory 118, which then communicates the information and instructions via bus, e.g.,
bus 127 of Figure 1C, to configurable processors, e.g., configurable modem processors 102a
and 102b and configurable codec processor 104. Within the configurable processors, local
20 memory, e.g., configuration memory 272, and local controller, e.g., controller 271 of Figures
2D and 2E, can control implementation of configuration information to, and operation of,
hardware satellite kernels, e.g., satellite kernel 270, in the present embodiment. Following
step 6102, flowchart 6100 proceeds to step 6104.

In step 6104 of the present embodiment, design configuration software (s/w)
25 is loaded into control registers. Step 6104 is implemented, in one embodiment, by storing
design configuration data in configuration information block 272 of the appropriate hardware
kernel. Card controller 110a or processor 112 of communication device 100 in Figure 1B can
direct the configuration information to the appropriate control register addresses within
configuration information block 272 of Figure 2D as identified by an off-line mapping
30 operation, e.g., a programming interface (or programming tools) on a separate workstation.
In this manner, a hardware kernel designed for integrate and dump functions will receive the
configuration software information related to integrate and dump functions.

Steps 6104 and 6102 can be implemented by receiving a computer program with data
and instructions for configuring the configurable electronic communication device.

35 Controllability, observability, and new configurations and behaviors can be realized via using

the extensible data types of the present invention. Thus the present invention overcomes limitations associated with the static prior art configuration for communication devices. The configuration data and computer program having instructions to implement the configuration data are implemented by a virtual machine interface (VMI), in the present embodiment, that quickly and efficiently translates the computer language information to the appropriate configuration mappings. Figure 1D provides an exemplary VMI 150 for interfacing configurable components, e.g., configurable modem processor 120a and configurable codec processor 140 of configurable communication device 100, as shown in Figure 1B. In particular, I/O device driver block 166 contains the respective drivers for the algorithmic-specific hardware kernels (or satellite kernels) described in Figures 2C through 2F. For example, if hardware kernel block K1 261a through K6 266a of Figure 2C have architectures that are designed for demodulation functions, then the necessary drivers for each of these hardware kernel blocks will exist in I/O device drivers block 166. These drivers will be used to configure the hardware kernels and implement the desired functions. Device drivers can exist for all the functions covered by a given application, e.g., for communication device 100 of Figure 1B. Figure 1D provides additional description of the VMI operation.

Following step 6104, flowchart 6100 proceeds to step 6106. In step 6106 of the present embodiment, an operating system interface is loaded into a controller. Step 6106 is implemented, in one embodiment, by receiving the operating system interface, as determined by a configuration design process. The controller receiving this information can range from: BTS card controller 110a of Figure 1B; BTS cell controller 114 of Figure 1C; allocator (controller) 219 of Figure 2A; and the individual hardware kernel controllers, e.g., controller 271 of Figure 2D, in each of the hardware kernel planes, e.g., plane [1] 201a through plane [i] 201i.

In particular, BTS channel card controller 110a (or an optional DSP) hosts the software stack developed by the user, e.g., per a configuration design process for a configurable device, that includes a standards-driven OSI Layer 1 Modem software stack. This software controls the operation of a modem signal processor, e.g. configurable modem processor 102a, and a channel codec signal processor, e.g., codec processor 104. This software stack exploits the full power of an application programming interface (API) for the wireless communication platform to realize efficient radio link performance for each channel

via a user's proprietary signal processing techniques. In this manner, the present embodiment effectively employs a hierarchy of controllers for providing configuration information and control information that will allow the autonomous operation of the individual hardware kernels. This allows for efficient parallel processing, with the benefit of reconfigurability.

Following step 6106, flowchart 6100 proceeds to step 6108. In step 6108 of the present embodiment, the operating system (OS) interface is linked with the API. Step 6108 is implemented in one embodiment by linking the appropriate set of interfaces from the API, e.g., from API inputs 172 of Figure 1D. The OS interface data and the API inputs can be stored in memory 118 on communication device. Step 6108 effectively reconciles the two interfaces such that the OS of the communication device can implement the functions desired on the appropriate hardware kernels, as provided in a configuration design process. Following step 6108, flowchart 6100 ends.

Referring now to Figure 6B, a flowchart 6200 of the process used to operate a configurable electronic communication device is shown, in accordance with one embodiment of the present invention. By using the flowchart embodiment of the present invention, an application with different protocols can be provided for in a configurable electronic device. Consequently, the present invention provides a method for designing efficient and intelligent flexibility into a configurable device to accommodate current differences and protocol as well as future protocol changes. Flowchart 6200 is implemented, in general, using the functional block diagram of Figure 1D and the hardware block diagrams of Figures 1A through 1C, and figures 2A through 2F.

Flowchart 6200 begins with step 6202. In step 6202 of the present embodiment, a signal is received at a configurable electronic device for processing. Step 6202 is implemented, in an uplink transmission from a mobile to a BTS, by a communication device, e.g., device 100 of Figure 1B, receiving a signal via wireless radio frequency (RF) transmission. The signal has time-divided packets therein, as appropriate for TDMA systems. And the sending and the receiving devices could be a base transceiver station (BTS), a mobile unit, or a test platform. Furthermore, the signal can include data information, control information, configuration information, and options for implementing configuration information. In particular, step 6202 is implemented in one embodiment by receiving signal on antenna 120 and communicated by bus 129 to communication device 100, as shown in

Figure 1B. In a downlink embodiment for step 6202, a MTSO traffic interface sends voice, data, and control data packets, via bus 128 to the transmit-data-pump path in BTS modem platform communication device 100 according to the configuration specified by the BTS cell controller 114 for that specific payload, as shown in Figure 1B. In an uplink embodiment for step 6202, the communication device 100, under control from the BTS card controller 110a, assigns a signal source to a receive-data-pump path in the modem signal processor. Following step 6202, flowchart 6200 proceeds to step 6204. In step 6204 of the present embodiment, the signal is preprocessed. Step 6204 is implemented, in one embodiment, by performing preprocessing steps that prepare a signal for the core processing to be performed by the reconfigurable hardware kernels of the communication device. For example, in a downlink scenario, one embodiment of step 6204 includes preprocessing steps such as disassembly of the signal and synchronization of the signal with over the air timing. Preprocessing can also include demuxing and sector-by-sector combining in the cases where channels from different sectors arrive from different modem signal processor transmit paths. In an uplink scenario, one embodiment of step 6204 includes parallel steps of synchronizing the signal and providing different types of data for the subsequent processing and transmission. By preprocessing the signal, it is prepared for the scope of processing available to a given hardware design kernel. Following step 6204, flowchart 6200 proceeds to step 6206.

In step 6206 of the present embodiment, a configuration for a configurable element is determined for a given user/channel. As mentioned in step 6104, the configuration can be user/channel specific. Thus, the configuration can include inputs such as radio type 6207a or quality of service (QOS) level 6207b, to determine the appropriate configuration of the user/channel, or to determine the specific option within an appropriate configuration. Radio type of service 6207a can include various TDMA communication protocols. QOS level 6207b can be a contracted level of reception quality or optional service features. While specific input types and choices are provided in the present embodiment, the present invention is well suited to using any type of input, and choice therein, as is supported by the design of the configurable device.

Step 6206 is implemented in one embodiment using memory 118, BTS card controller 110b of communication device 101 in Figure 1C, or by using allocator 219 of Figure 2A and

configuration information block 272 of Figure 2D. In one embodiment, a physical channel can be tied to a specified configuration, and thus established a priori. Alternatively, the data to be processed by the configurable device can contain header information that indicates a protocol, and hence an appropriate configuration for the hardware kernels assigned to process that protocol.

Following step 6206, flowchart 6200 proceeds to step 6208. In step 6208 of the present embodiment, the signal is assigned a data pump path on one or more independent processors. Step 6208 is implemented, in one embodiment, by allocator 219 and/or microprocessor 112 of Figure 2A, in conjunction with memory. Microprocessor 112 and allocator 219 use a unique set of communication primitives to indicate the data pump path between multiple hardware kernels of which the data is required to flow for data processing that will satiate the desired functions. The information needed for a data pump path includes the hardware kernel addresses from which, and to which, data flows for the required data processing operation. The information also includes any configuration data necessary for the reconfigurable interconnect, e.g., interconnect 204a of Figure 2C, to realize the interconnection of the two or more hardware kernels. The address, instructions, and configuration information (such as reconfigurable interconnect configuration) are stored in memory 118 of communication device 100 of Figure 1B and/or memory portion of configuration information block 272 or in controller block 271 of a given hardware kernel as shown in Figures 2D and 2E.

As an example related to step 6208, several function blocks, e.g., for function group B 316b of Figure 3D, can be implemented on hardware kernel plane 201a of Figure 2C. In particular, hardware kernel group A 268a of Figure 2C can utilize hardware kernels K1 261a, K4 264a, and K5 265a, with the appropriate interconnects between them being established by reconfigurable interconnect 204a to implement a given function, or group of functions. The data pump path refers to the hardware kernels used to implement the functions, and the sequence of data flow between these appropriate kernels. Following step 6208, flowchart 6200 proceeds to step 6210.

In step 6210 of the present embodiment, design configuration information is received at a configurable hardware kernel for processing a respective portion of the signal. Step 6210 is implemented, in one embodiment, by receiving design configuration information from

configuration information block 272 at satellite kernel 270, within the hardware kernel 261 a, as shown in Figure 2D. Alternatively, the memory location of the information can be provided elsewhere, such as a memory portion of allocator 219. This information is downloaded in the present embodiment of the device, following an off power condition that cleared memory in satellite kernel 270. Alternatively, this information can be reloaded for an interrupt or fault condition, as preset or determined by a user. New configuration information can be received while the device is powered up, thus overriding the previous configuration. Functional block diagram 200f shows that control signal generation block 284 provides a configuration information function via communicating the configuration to algorithmic computation block 292 via line 295d. Following step 6210, flowchart 6200 proceeds to step 6212.

In step 6212 of the present embodiment, an inquiry determines if timesharing is occurring. If time-sharing is occurring, then flowchart 6200 proceeds to step 6214. However, if timesharing is not occurring, then flowchart 6200 skips to step 6216. Step 6212 provides the logic to determine whether a configurable device has a timeshare set up or not. The timeshare set up requires additional management and memory storage steps, as indicated hereinafter. A configuration input 121 to a communication device can indicate whether time-sharing is used or not, and to what extent it is used. Step 6212 is implemented, in one embodiment, via allocator 219 or processor 112 of Figure 2A. In particular, allocator 219 or processor 112 can be used to determine whether time-sharing exists, e.g., via preprogrammed timeshare configuration of a channel ID or some other identifier. Furthermore, a configuration design process provides the necessary configuration information and management logistics to implement a timesharing set up.

Time-sharing, per step 6212, can exist on a wide range of hardware levels. For example, time-sharing can exist on a hardware kernel by hardware kernel basis, e.g., K1 261a, or on a hardware plane by hardware plane basis, e.g. hardware kernel plane 201a. Furthermore, timesharing can occur intermittently on a device, as programmed by a user. For example, a device can time-share in terms of temporal variation, e.g., different parts of the day require more or less resources, or spatial diversity, and e.g., some sectors of a base station having higher resource requirements than other sectors. By providing scaled clock speeds to discrete hardware, e.g., hardware kernels, and by providing a selectable quantity of resources,

5 e.g., hardware kernels, the present invention provides effective scalability without significant changes to the architecture. Additional information on timesharing apparatus and processes is described in co-pending patent application entitled "IMPROVED APPARATUS AND METHOD FOR MULTI-THREADED SIGNAL PROCESSING," Serial No. 09/492,634, and filed on January 27, 2000. This related application is commonly assigned, and is hereby
10 incorporated by reference.

Step 6214 arises if time-sharing is desired, per step 6212. In step 6214 of the present embodiment, the state information for the applicable time slice is received. Step 6214 is implemented, in one embodiment, by retrieving a stored state of configurable computation kernel (or configurable satellite element) 273a in a memory portion of controller 271, or in
15 configuration information block 272, of Figures 2E or 2D respectively. Furthermore, the state of the reconfigurable interconnect 204a of Figure 2C can be stored in memory in allocator 219, as shown in Figure 2A, or in memory 118 of communication device 100, in another embodiment. The state information is provided to configurable computing element 273a. In particular, the type of state information is that required for performing the operation
20 for a given function. Thus, in one embodiment, the states in registers used for filtering can be stored. Likewise, the state of the data being processed can be stored, if the operation was not complete. Following step 6214, flowchart 6200 proceeds to step 6216.

In step 6216 of the present embodiment, the signal is processed by the appropriate configurable elements, e.g., the hardware plane or hardware kernel, etc., as shown in Figures
25 2A through 2F. In practice, after the API functions are executed on the processor per step 6204, control is transferred, e.g., via a handshake protocol, to the pool of reconfigurable kernels, e.g., K1 261a through K6 266a of Figure 2C. By transferring control, a lower level of controller now has control within the controller hierarchy system.

Step 6216 is implemented in the present embodiment on one or more hardware
30 kernels, e.g., kernel K1 261a, which has autonomous localized control to control an operation to completion. In this manner, the present embodiment provides a data processing island that conserves system resources by performing its operations locally, in a reconfigurable and/or time-shared manner. Step 6216 is enabled, in one embodiment, by allocator 219 of Figure 2A directing the appropriate data signal and control signal to the configurable computation
35 kernel, e.g., configurable computation kernel 273a, as described in "data pump assignment"

5 step 6208. Figure 2f shows the input/output interlaces and functions that enable step 6216. For example, input data 290 is received at algorithmic computation block 292, along with state information 295b, and other types of data such as adapted clock signal 295a and enable/status 295c that enable the algorithmic computation block. The interfaces and functions described in Figure 2f are implemented in hardware in Figures 2A through 2F.

10 In the present embodiment, the functions implemented in step 6216 for communication device 100 of Figure 1B are the modem functions 6216a and the codec functions 6217b. In particular, modem functions for a downlink scenario include encoding, interleaving, channelization code mixing, I/Q scrambling code mixing, transmit diversity processing, filtering, and rate-dependent scaling. Modem functions for uplink include:
15 despread, dechannelization, demodulation, and combining, as programmed by the Layer I software stack, including facilities to perform multipath and antenna diversity combining.

In contrast, the codec functions implemented for step 6216 for a downlink scenario include: encoding, interleaving, channelization code mixing, and I/Q scrambling code mixing, transmit diversity processing, filtering, and rate-dependent scaling. The codec functions for
20 uplink scenario include despread, deinterleaving, channel decoding, assembly of payload data and control information for MTSO. The present invention is well suited to implementing a wide range of functions in a hardware kernel, suitable for a given application. The modem signal processor, under control from the BTS Card Controller, assigns a signal source to a receive-data-pump path in the modem signal processor.

25 Following step 6216, flowchart 6200 proceeds to step 6218. In step 6218 of the present embodiment, the signal is post processed. Thus, after the pool of reconfigurable kernels, e.g., K1 261a through K6 266a of Figure 2C, have completed their operations on the data, then control, or computations, are returned back to the processor. Consequently, this completes the circle of hierarchical controls implemented by a return handshake. Step 6218
30 is implemented, in one embodiment, providing signal to subsequent downstream or upstream functions and devices that prepare the signal for the application, e.g., transmission of a signal via antenna, amplification of a signal for reception, etc. In particular, post processing for a downlink scenario from a BTS can include generating multiplexed I/Q digital base band waveforms identified with a channel, sector, and/or antenna tag. Another post processing
35 step includes formatting the composite signal, consisting of all the signals on a per-carrier and

5 per-sector, as necessary and sent from the channel card to other signal conditioning circuits in the base-station for digital to analog (D/A) conversion and for radio frequency (RF) transmission. Following step 6218, flowchart 6200 ends.

The present embodiment applies flowcharts 6100 and 6200 to a digital wireless communication system. However, the present invention can be applied to a wide range of applications and a wide range of device configurations. Within the wireless communication system described in the present embodiment, the present invention is applicable to mobile units, base stations, and test platforms.

While flowcharts 6100 and 6200 of the present embodiment show a specific sequence and quantity of steps, the present invention is suitable to alternative embodiments. For example, not all the steps provided in flowcharts 6100 and 6200 are required for the present invention. In particular, flowchart 5200 provides steps 5212 and 5214 for a timeshare scenario for hardware kernels. However, the time-scenario is optional for the present invention, and thus, these steps may be omitted in one embodiment. Similarly, other steps may be omitted depending upon the application. In contrast, the present invention is well suited to incorporating additional steps to those presented, as required by an application, or as desired for permutations in the process.

Lastly, the sequence of the steps for flowcharts 6100, and 6200 can be modified depending upon the application. Thus, while flowcharts 6100 and 6200 are shown as a single serial process, they can also be implemented as a continuous or parallel process. For example, is appreciated that flowchart 6100 can be repeated for the multiple hardware kernel planes, e.g., plane 201a of Figure 2C, in the multiple processors, e.g., processors 102a and 102b of Figure 1B. within a communication device, e.g., device 100.

Many of the instructions for the steps, and the data input and output from the steps, of flowcharts 6100, and 6200 utilize memory and processor hardware components, e.g., memory 118 processor 112 of Figure 1B. The memory storage used to implement the flowchart steps in the present embodiment can either be permanent, such as read only memory (ROM), or temporary memory such as random access memory (RAM). Memory storage can also be any other type of memory storage, capable of containing program instructions, such as a hard drive, a CD ROM, or flash memory. Similarly, the processor used to implement the flowchart steps can either be a dedicated controller, an existing system processor, or it can be

5 a dedicated digital signal processing (DSP) processor, as appropriate for the type of step.
Alternatively, the instructions may be implemented using some form of a state machine.
Some portions of the detailed description, e.g., the processes, are presented in terms of
procedures, logic blocks, processing, and other symbolic representations of operations on data
bits within a computer or digital system memory or on signals within a communication
10 device. These descriptions and representations are the means used by those skilled in the
digital communication arts to most effectively convey the substance of their work to others
skilled in the art. A procedure, logic block, process, etc., is herein, and generally, conceived
to be a self-consistent sequence of steps or instructions leading to a desired result. The steps
are those requiring physical manipulations of physical quantities. Usually, though not
15 necessarily, these physical manipulations take the form of electrical or magnetic signals
capable of being stored, transferred, combined, compared, and otherwise manipulated in a
communication device or a processor. For reasons of convenience, and with reference to
common usage, these signals are referred to as bits, values, elements, symbols, characters,
terms, numbers, or the like with reference to the present invention.

20 It should be borne in mind, however, that all of these terms are to be interpreted as
referencing physical manipulations and quantities and are merely convenient labels to be
interpreted further in view of terms commonly used in the art. Unless specifically stated
otherwise as apparent from the following discussions, it is understood that throughout
discussions of the present invention, terms such as "receiving," "generating," "mapping,"
25 "repeating," "identifying," "translating," "dividing," "decoding," "defining," "time-sharing,"
"scheduling," "assigning," "creating," "categorizing," "loading," "interfacing,"
"disassembling," "performing," "synchronizing," "demuxing," "transmitting," "combining,"
"formatting," "assembling," or the like, refer to the action and processes of a communication
device or a similar electronic computing device, that manipulates and transforms data. The
30 data is represented as physical (electronic) quantities within the communication devices
components, or the computer system's registers and memories, and is transformed into other
data similarly represented as physical quantities within the communication device
components, or computer system memories or registers, or other such information storage,
transmission or display devices.

35 In view of the embodiments presented herein, the present invention effectively

5 provides a method and apparatus that can effectively accommodate the increases in the quantity of users and the quantity of data transferred using the limited frequency bandwidth. Furthermore the present invention provides a solution that overcomes the limitations of protocol non-uniformity and proliferation in the wireless communications. The limitations of cost and burden associated with changes in versions or performance levels of a communication protocol are also resolved by the present invention. In an effort to minimize the risks and maximize the rewards, the present invention substantially shortens the lead-time and the investment required for implementing a new specification. Finally, the present invention provides very reasonable power consumption for the communication device.

15 The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and naturally, many modifications and variations are feasible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable those skilled in the art to best utilize the invention and various embodiments with various modifications as is suitable to the particular use contemplated. It is intended that the scope of the invention be defined by the claims appended hereto and their equivalents.